AAI/CPE/EE 595-WS 1
Applied Machine Learning

Lecture 4: Decision Tree

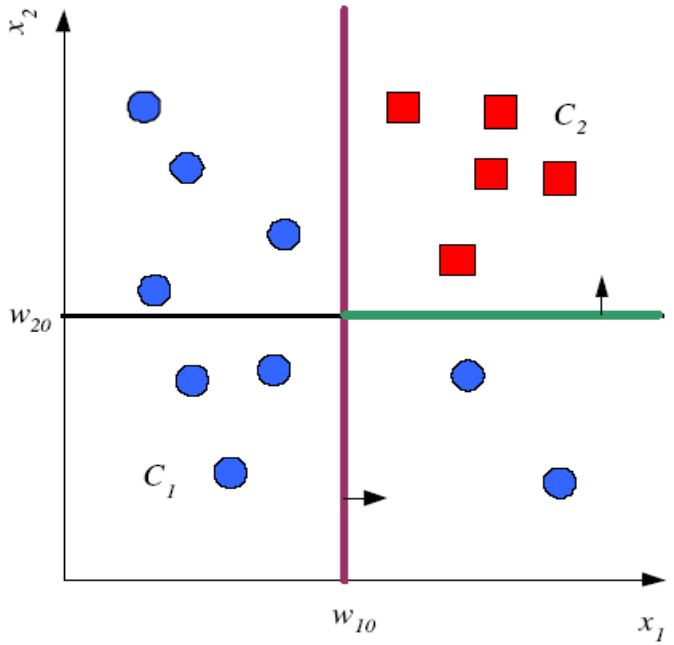
Speaker: Jiarui Li
jli148@stevens.edu

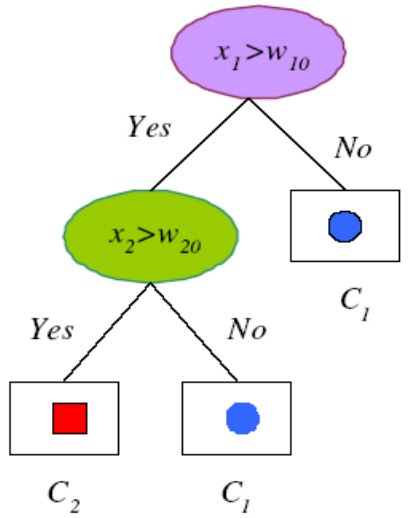
Decision Tree

- A hierarchical structure that used to classify classes based on a set of rules
  - called ***classification tree***, if the target values are discrete
  - called regression tree, if the target values are continuous

- Training decision trees is to iteratively split each node at optimum attribute based on ***impurity*** or ***entropy*** measures

- Training process is greedy





root

Internal Node

Leaf

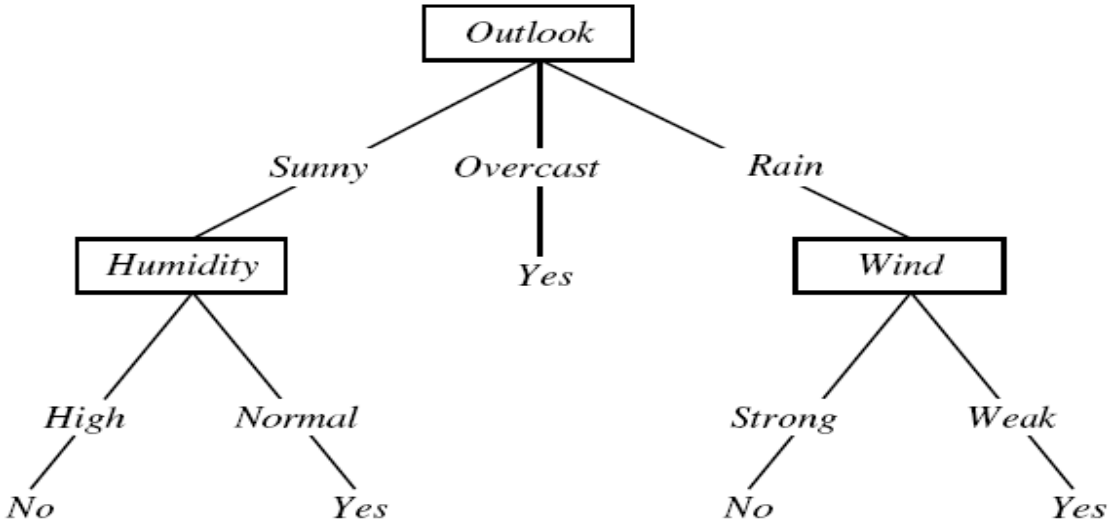
Training Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

Decision Tree Elements

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

Decision tree for *PlayTennis*



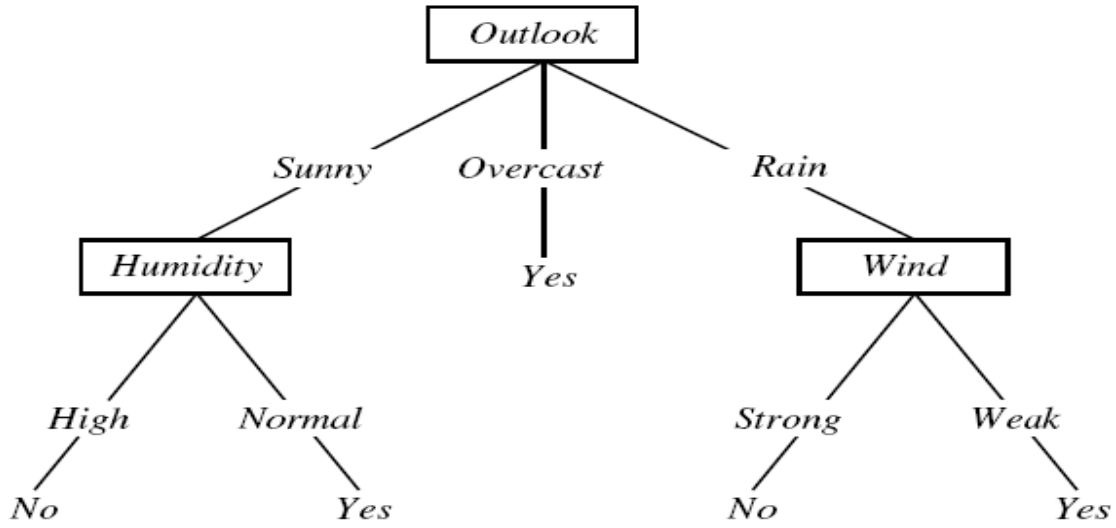
Popular classification tree algorithms: ID3, CART, ASSISTANT, C4.5,…

Decision Tree

- A decision tree represents a **disjunction of conjunctions** of constraints on the attribute values of instances

When do you play Tennis?

(Outlook = Sunny Λ
Humidity = Normal) V
(Outlook = Overcast) V
(Outlook = Rain Λ Wind =
Weak)



All decision trees make a **complete space** of finite discrete-valued functions.

Where decision tree learning is appropriate

- For problems with the following characteristics:
  - Instances are represented by attribute-value pairs
    - Real-valued attributes are fine
  - Target functions has discrete output values
    - Real-valued outputs are fine
  - Disjunctive descriptions may be required
  - The training data may contain errors
  - The training data may contain missing attribute values
  - **Interpretability**

Training Decision Trees

- Popular Algorithms:
  - ID3, C4.5, CART (classification & regression tree)

- Training in the **top-down** manner, starting from the root node
  - At current node, choose the best attribute based on certain criterion
    - The ID3 algorithm use ***information gain*** to determine attributes
    - The CART algorithm uses ***Gini impurity***
  - For each value of the selected attribute, create a child node
  - Sort the training examples to the leaf nodes
  - Repeat above steps until training examples are perfectly classified or no new attribute is available

Entropy

- The Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

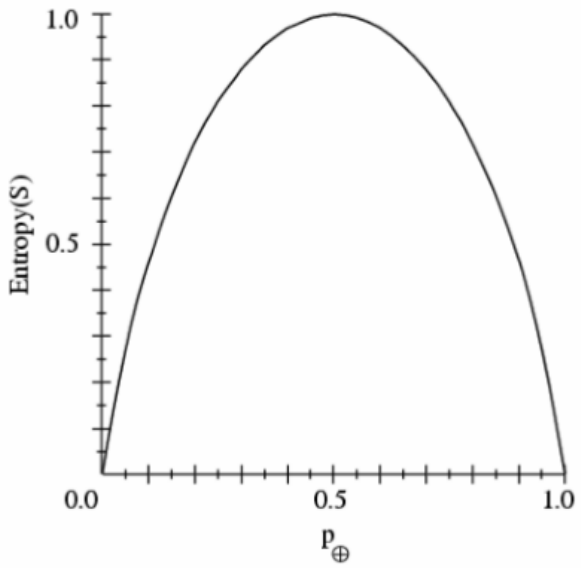
We have n possible outcomes (classes), $P(X = i)$ is the proportion of instances belonging to class I
The logarithm is base 2, which reflects the binary nature of the decision-making

Why? In Information Theory:

- Most efficient code assigns $-\log P(X = i)$ bits to encode the message $X = i$
- Thus, expected number of bits is

$$\sum_{i=1}^{n} P(X = i)(-\log_2 P(X = i))$$

Sample Entropy



- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Define $0\log_2 0 = 0$

Information gain

- Measures the reduction in entropy after the dataset is split on an attribute. In other words, it tells us how much uncertainty is reduced when we use a specific attribute to split the data.
- The more a feature reduces uncertainty (i.e., the greater the information gain), the better that feature is for classification.

- Let *S* be a set of examples, and *A* is an attribute, the entropy of set *S* is *H(S).*
- We split S into subsets S1, s2…,sk based on the values of attribute A, then the entropy after splitting on A is the weighted sum of the entropy of each subset:

$$H(S|A) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} H(S_i)$$

- Then the information gain is calculated as

$$G(A) = H(S) - H(S|A)$$

Specific Conditional Entropy:

$H(S\,|A = vi)$ = The entropy of S among the records in which *A* has value v_i

Conditional Entropy:

$H(S\,|A)$ = The average specific conditional entropy of $S = \sum_i \Pr[(A = vi)\, H(S \mid A = vi)]$

Information gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Specific Conditional Entropy:

$H(S \mid A = vi)$ = The entropy of S among the records in which A has value v_i

Conditional Entropy:

$H(S \mid A)$ = The average specific conditional entropy of $S = \sum_i \Pr[(A = vi)\, H(S \mid A = vi)]$

Training Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

Calculate *H(S)*

- We have 14 total instances, 9 Yes and 5 No
- The initial entropy of the dataset S is calculated as follows.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

$$H(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$H(S) \approx -\left(\frac{9}{14}\right) \times 0.514 - \left(\frac{5}{14}\right) \times 0.737$$

$$H(S) \approx -0.330 - 0.264$$

$$H(S) \approx 0.940$$

Calculate *Entropy for Each Branch*

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

- Let's split by outlook
- Outlook = Sunny (S_Sunny)
- We have five total instances, D1, D2, D8, D9, D11
- Classes: 2 Yes, 3 No
- Let's calculate the entropy

$$H(S_{\text{Sunny}}) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) \qquad H(S_{\text{Sunny}}) \approx -0.528 - 0.442 = 0.971$$

- Similarly

$$H(S_{\text{Overcast}}) = -\left(\frac{4}{4}\right)\log_2\left(\frac{4}{4}\right) - \left(\frac{0}{4}\right)\log_2\left(\frac{0}{4}\right) = 0$$

$$H(S_{\text{Rain}}) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) \qquad H(S_{\text{Rain}}) \approx -0.442 - 0.528 = 0.971$$

Entropy After Splitting with ‘Outlook’

- Let’s calculate the weighted conditional entropy

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

$$H(S \mid \text{Outlook}) = \frac{5}{14} \times H(S_{\text{Sunny}}) + \frac{4}{14} \times H(S_{\text{Overcast}}) + \frac{5}{14} \times H(S_{\text{Rain}})$$

$$H(S \mid \text{Outlook}) = \frac{5}{14} \times 0.971 + \frac{5}{14} \times 0.971$$

$$H(S \mid \text{Outlook}) = \frac{10}{14} \times 0.971$$

$$H(S \mid \text{Outlook}) \approx 0.693$$

Information Gain

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

- Let's calculate the information gain for the 'outlook' attribute

$$IG(\text{Outlook}) = H(S) - H(S \mid \text{Outlook})$$

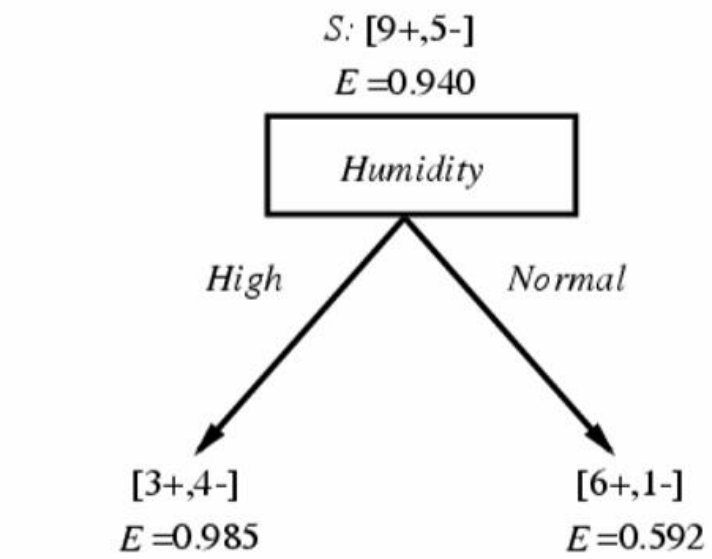
$$IG(\text{Outlook}) = 0.940 - 0.693$$

$$IG(\text{Outlook}) \approx 0.247$$

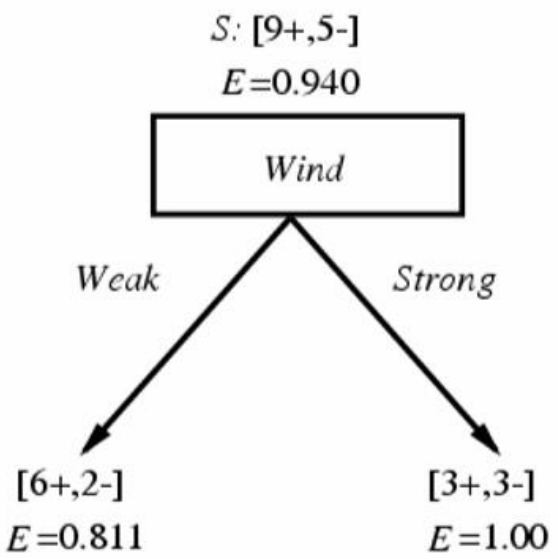
Attribute choosing

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

- Which attribute is the best classifier?



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

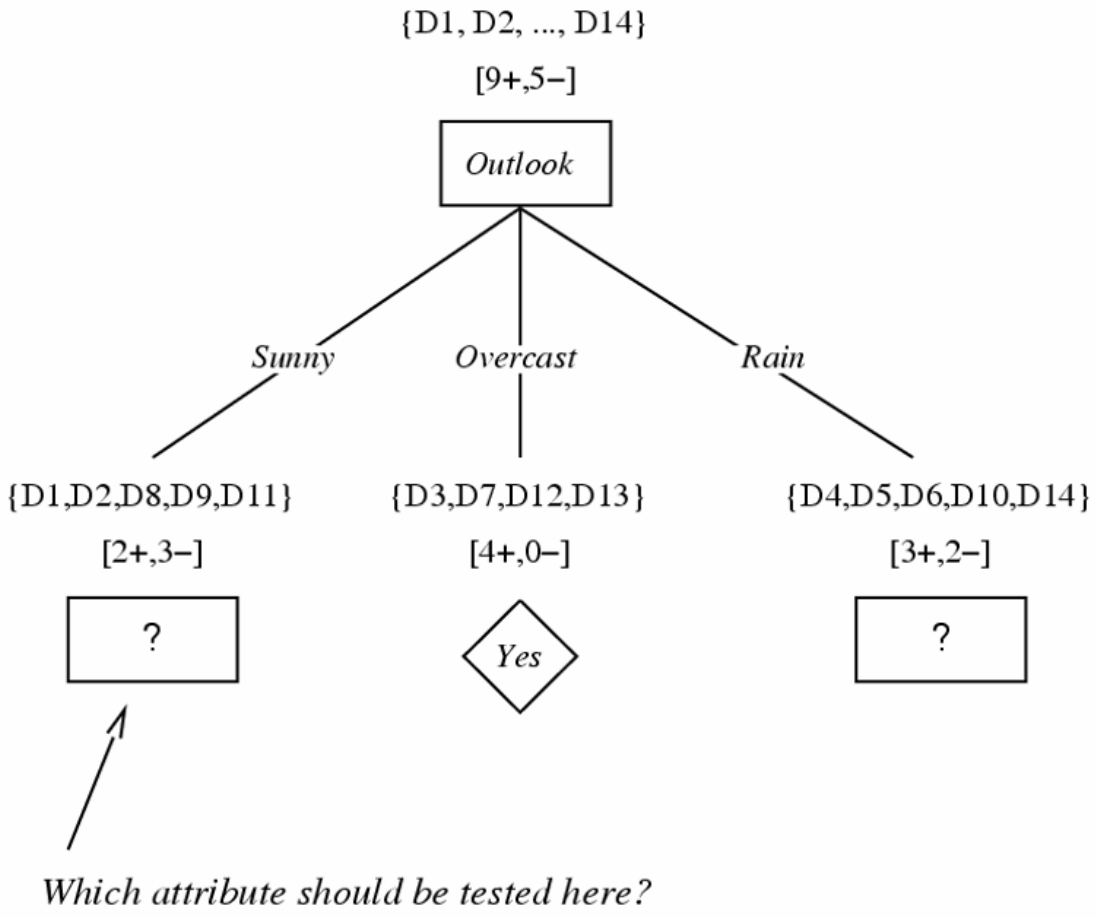


Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

Gain(S, Outlook) = 0.247
Gain(S, Temperature) = 0.029

Attribute choosing

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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



S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\,(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain\,(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain\,(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Coding Example (1/3)

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

```python
# Function to calculate entropy
def entropy(class_labels):
    total = len(class_labels)
    class_counts = {}

    for label in class_labels:
        if label not in class_counts:
            class_counts[label] = 0
        class_counts[label] += 1

    entropy_value = 0.0
    for count in class_counts.values():
        probability = count / total
        entropy_value -= probability * math.log2(probability)

    return entropy_value
```

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

```python
# Function to calculate information gain
def information_gain(S, subsets):
    # Entropy of the original set
    entropy_S = entropy(S)

    # Weighted entropy of the subsets
    total_len = sum(len(subset) for subset in subsets)
    weighted_entropy = sum((len(subset)
                            / total_len) * entropy(subset) for subset in subsets)

    # Information gain is the difference in entropy
    gain = entropy_S - weighted_entropy
    return gain
```

Coding Example (2/3)

- Create the dataset and split the input attributes and output
- Calculate the total entropy and gains for each attribute

```
# Calculate entropy of the entire dataset
entropy_total = entropy(labels)
print(f'Entropy of the entire dataset: {entropy_total:.3f}')

# Splitting based on the 'Outlook' attribute
sunny_subset = [row[-1] for row in dataset if row[0] == 'Sunny']
overcast_subset = [row[-1] for row in dataset if row[0] == 'Overcast']
rain_subset = [row[-1] for row in dataset if row[0] == 'Rain']

# Calculate information gain for the 'Outlook' attribute
gain_outlook = information_gain(labels, [sunny_subset, overcast_subset, rain_subset])
print(f'Information Gain for Outlook: {gain_outlook:.3f}')

# Further splits for 'Sunny' based on 'Humidity'
sunny_high_humidity = [row[-1] for row in dataset if row[0] == 'Sunny' and row[2] == 'High']
sunny_normal_humidity = [row[-1] for row in dataset if row[0] == 'Sunny' and row[2] == 'Normal']

# Calculate information gain for 'Humidity' within the 'Sunny' subset
gain_sunny_humidity = information_gain(sunny_subset, [sunny_high_humidity, sunny_normal_humidity])
print(f'Information Gain for Humidity within Sunny: {gain_sunny_humidity:.3f}')

# Further splits for 'Rain' based on 'Wind'
rain_weak_wind = [row[-1] for row in dataset if row[0] == 'Rain' and row[3] == 'Weak']
rain_strong_wind = [row[-1] for row in dataset if row[0] == 'Rain' and row[3] == 'Strong']

# Calculate information gain for 'Wind' within the 'Rain' subset
gain_rain_wind = information_gain(rain_subset, [rain_weak_wind, rain_strong_wind])
print(f'Information Gain for Wind within Rain: {gain_rain_wind:.3f}')
```

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

Output:

Entropy of the entire dataset: 0.940
Information Gain for Outlook: 0.247
Information Gain for Humidity within Sunny: 0.971
Information Gain for Wind within Rain: 0.971

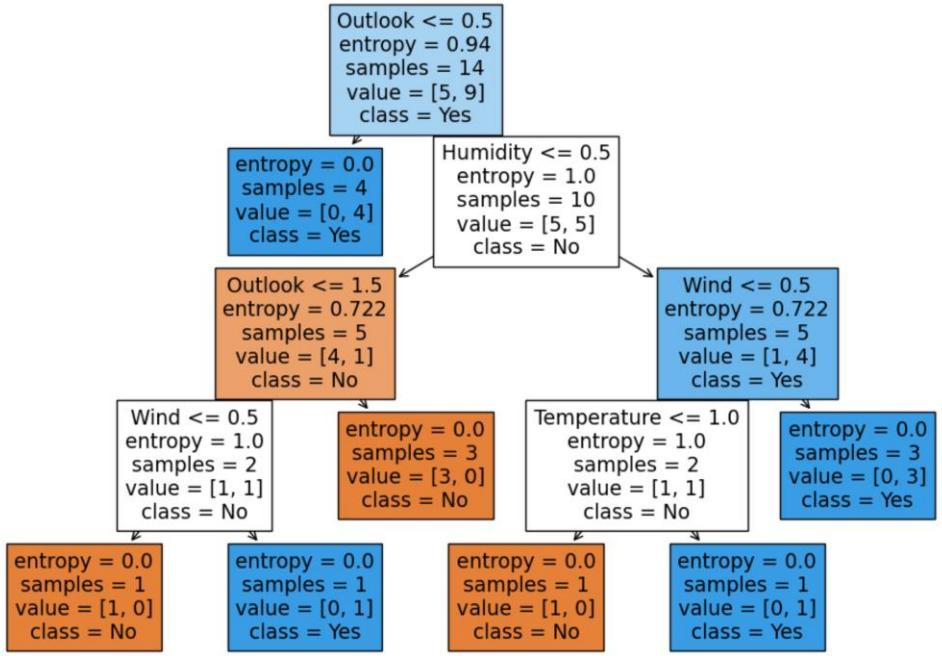
Coding Example (3/3)

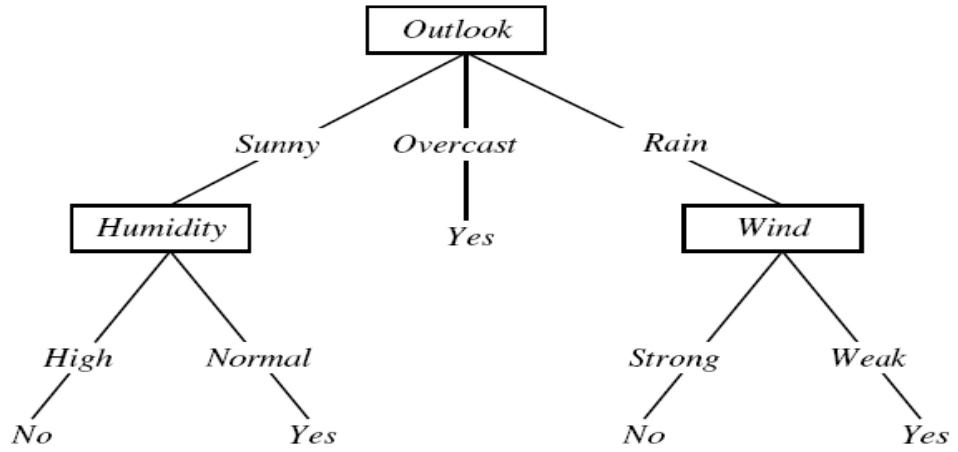
- Display the tree using DecisionTreeClassifier

```
# Step 4: Separate features (X) and target (y)
X = df[['Outlook', 'Temperature', 'Humidity', 'Wind']]
y = df['PlayTennis']

# Step 5: Create and train the decision tree classifier
clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
clf.fit(X, y)
```

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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





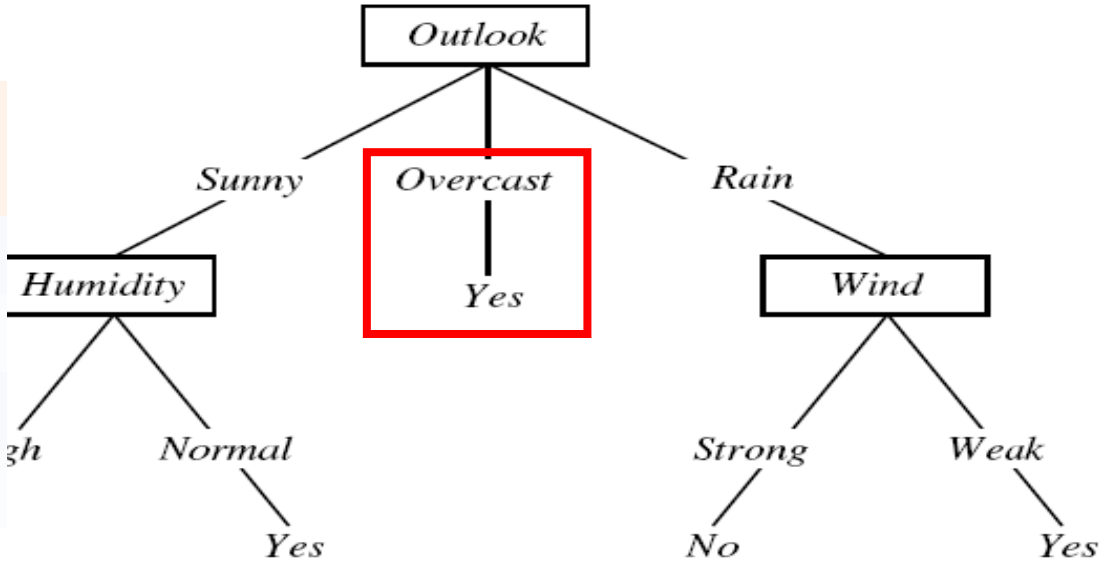
Stop Criteria

1. Every attribute has already been included along the path

2. Training examples associated with this node ALL have the same target attribute value (i.e., their entropy is zero).

3. Don’t split a node if none of the attributes can create multiple non-empty children

Age	Income	Has credit card?	Buys Product
25	High	Yes	Yes
25	High	Yes	Yes
25	High	Yes	Yes
25	High	Yes	Yes



Overfitting

- Definition: If your machine learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is ***overfitting***.

- the model performs exceptionally well on the training data but generalizes poorly to unseen data (test data).

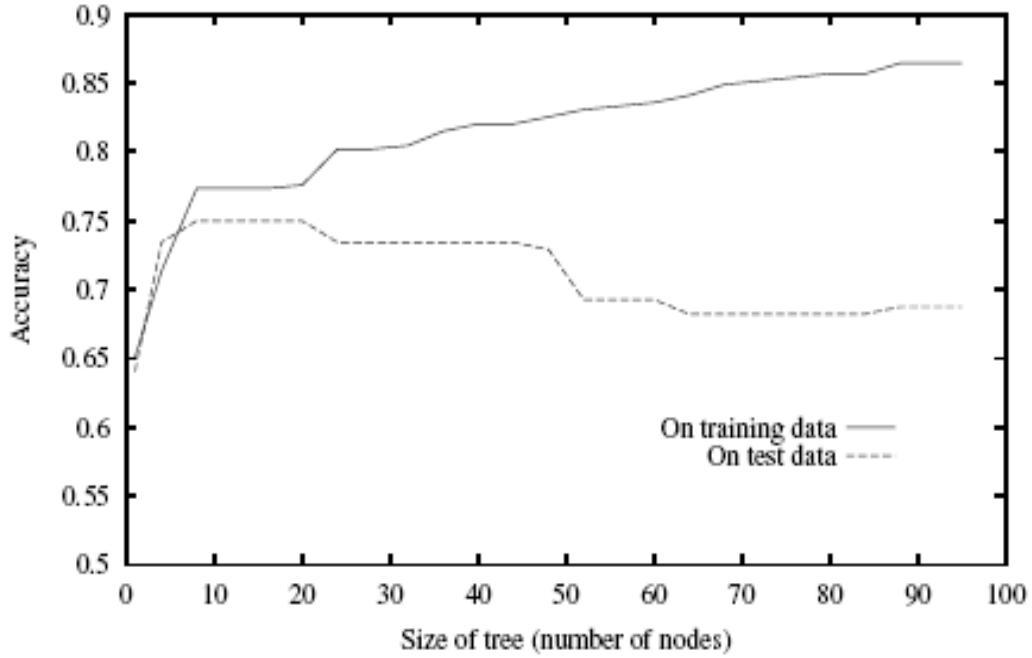
- If we have a hypothesis h and it has two types of error $error_{train}(h)$ and $error_{test}(h)$

- Underfitting: $error_{train}(h)$ is high and $error_{test}(h)$ is high

- Good fit: $error_{train}(h)$ is low and $error_{test}(h)$ is low

- Overfitting: $error_{train}(h)$ is very low and $error_{test}(h)$ is high

Overfitting in Decision Tree – ID3 algorithm



Observe overfitting:

1) Test data error is larger and larger

2) Training data error is small

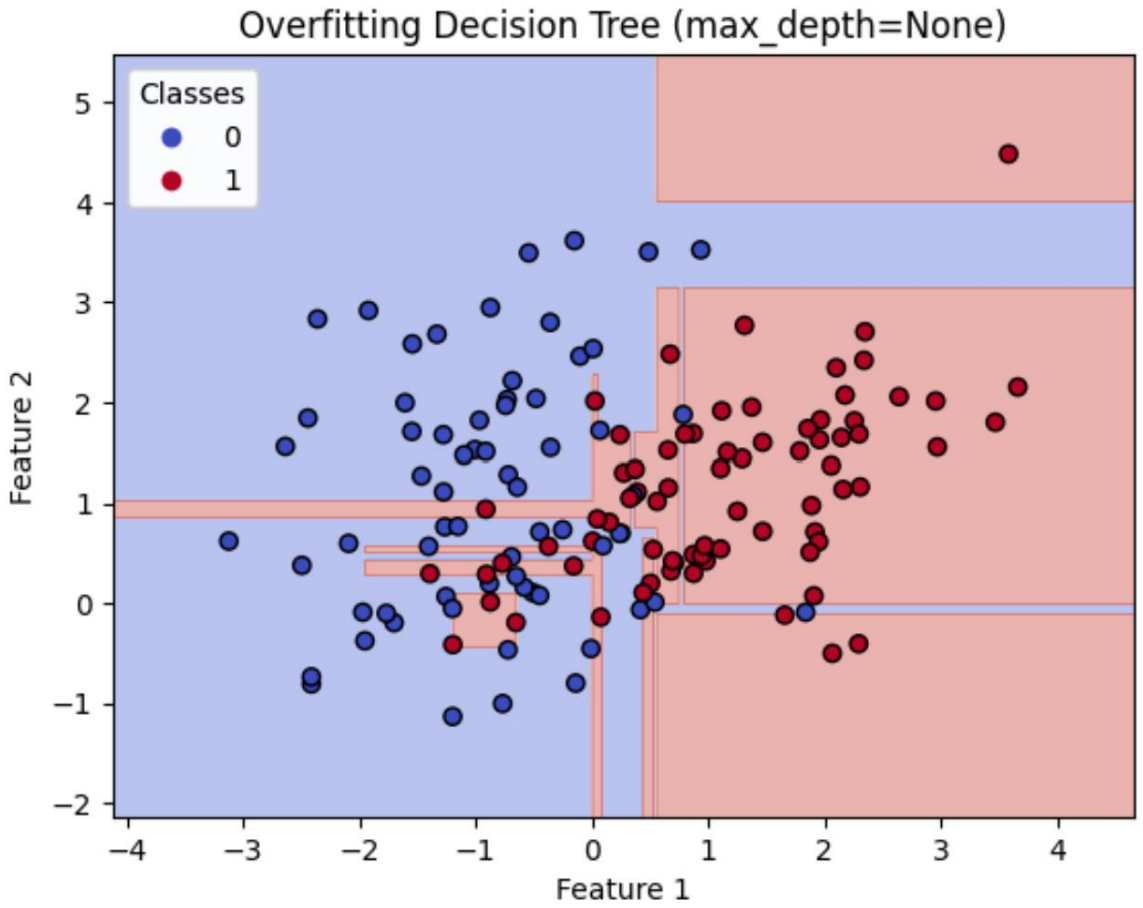
Bigger trees more likely overfits!

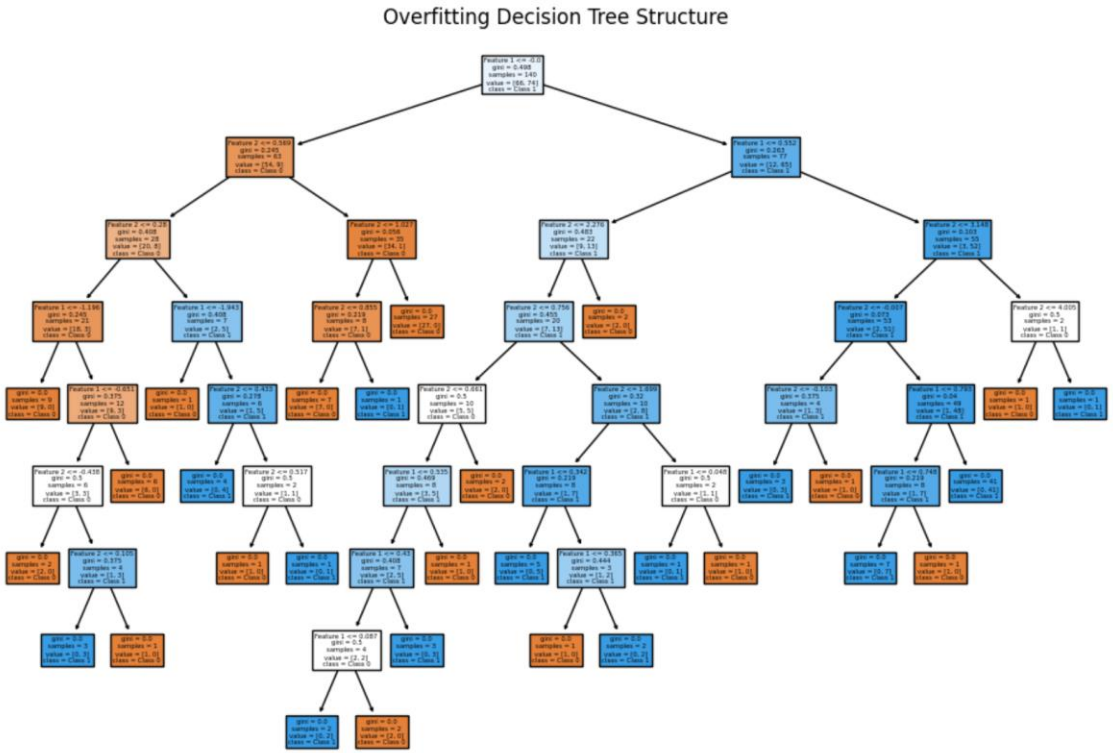
Reduce overfitting:

1) Stop growing the tree when data split is not statistically significant.

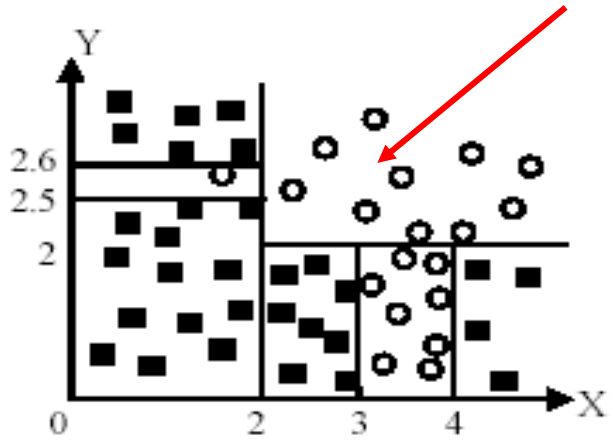
2) Grow a full tree, then prune it

Overfitting Example

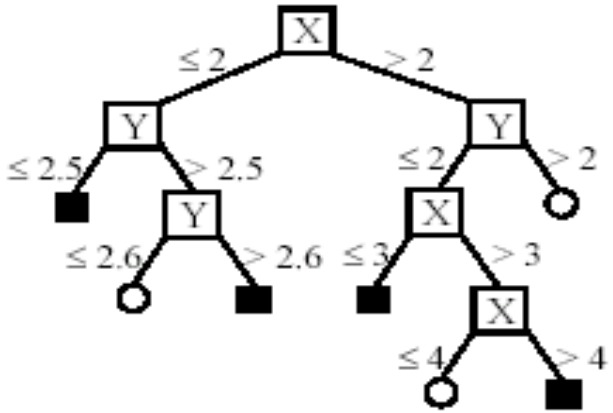




Overfitting Example (cont.)

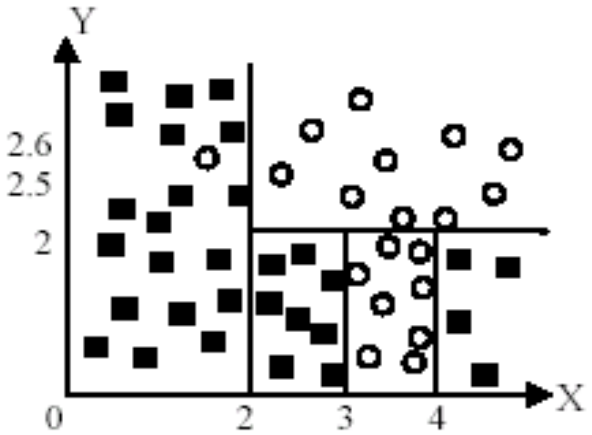


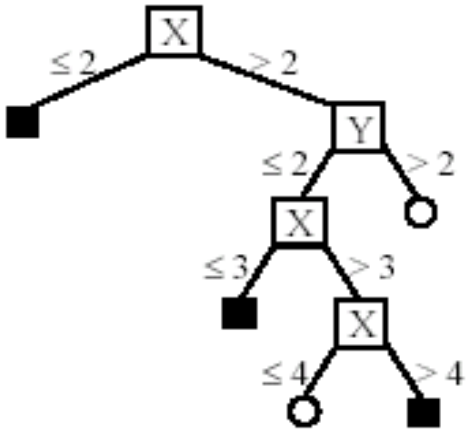
(A) A partition of the data space



(B). The decision tree

Likely to overfit the data





Pruning Strategy

- **Pre-pruning** (early stopping): stop growing when data split not statistically significant
  - tuning the hyperparameters: ‘max_depth’, ‘min_samples_leaf’, ‘min_sample_split’
  - Use ***cross-validation*** to find the best hyperparameters
  - Notice: pre-pruning might lead to underfitting since it may prevent some meaningful split
- **Post-pruning**: Allow the model to grow to its full depth and then remove the branches to prevent overfitting
  - Reduced Error Pruning (REP): substitute subtree with a leaf node and compare the performance
  - Pessimistic Error Pruning (PEP): Remove the branch that brings the lowest error among all possible removal
  - Cost Complexity: select a subtree to prune such that this pruning leads to a lower test error
  - Chi-Square Test

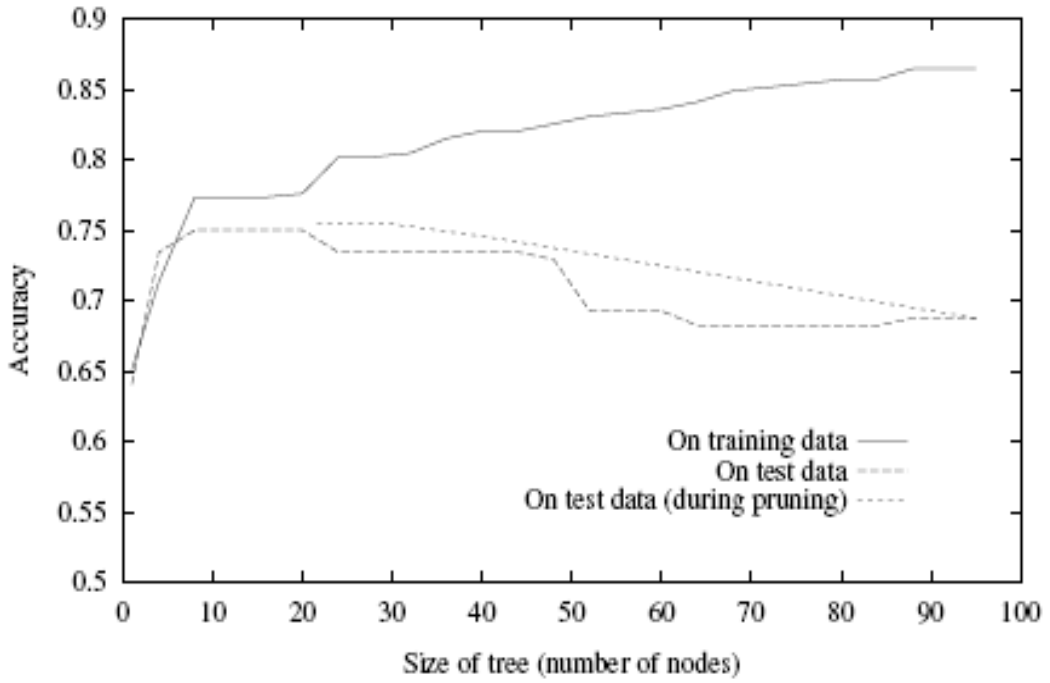
Reduced error pruning

Split data into **training data** and **validation data**.

Repeat the following until further pruning is harmful:

- Create a set of different trees by pruning each possible node and the subtree below it
- Evaluate the validate set accuracy of each tree
- Keep the one that most improves the validation set accuracy

This approach produces the smallest and most accurate tree. But it is computationally costly & requires sufficient training data.



Pessimistic Error Pruning

Removes branches from the tree based on a pessimistic estimate of the error.

This technique does not require a separate validation set.

Repeat the following until further pruning would increase the pessimistic error estimate.:

- For each node or subtree, estimate the error using the training data. This error is adjusted pessimistically (to account for the likelihood of overfitting).
- For each branch in the decision tree, calculate the error that would result if the branch was removed and replaced with a leaf node.
- Compare all possible branches and remove the one that results in the lowest pessimistically estimated error.

Pessimistic error is estimated as follows

$$\text{Pessimistic Error} = \text{Training Error} + k \times \frac{L}{N}$$

L is the number of leaf nodes in the subtree

N is the total number of instances in the training set

k is a constant (often 0.5), reflecting the assumed increase in error.

Pessimistic Error Pruning (cont.)

- Assume: $e(t) = \#\ of\ error\ samples, n(t) = total\ \#$ of nodes. In general, the error rate of a node can be represented by $err_t = \frac{e(t)}{n(t)}$, the error rate for subtree T_t, $err_{T_t} = \frac{\Sigma e(s)}{\Sigma n(s)}, s \in T_t$

In PEP, we consider an additional error with penalty 0.5

we set $err'_t = \frac{e(t)+0.5}{n(t)}$, $err'_{T_t} = \frac{\Sigma(e(s)+0.5)}{\Sigma n(s)}, s \in T_t, s\ is\ leaf$

The standard deviation of error is: $std(err'_{T_t}) = \sqrt{n(t) \cdot err'_{T_t} \cdot (1 - err'_{T_t})}$ (binomial dist.)

Pruning the subtree T_t if for node t, we have:

$$e(t) \leq n_t \cdot err'_{T_t} + std(err'_{T_t})$$

An Example (Dataset)

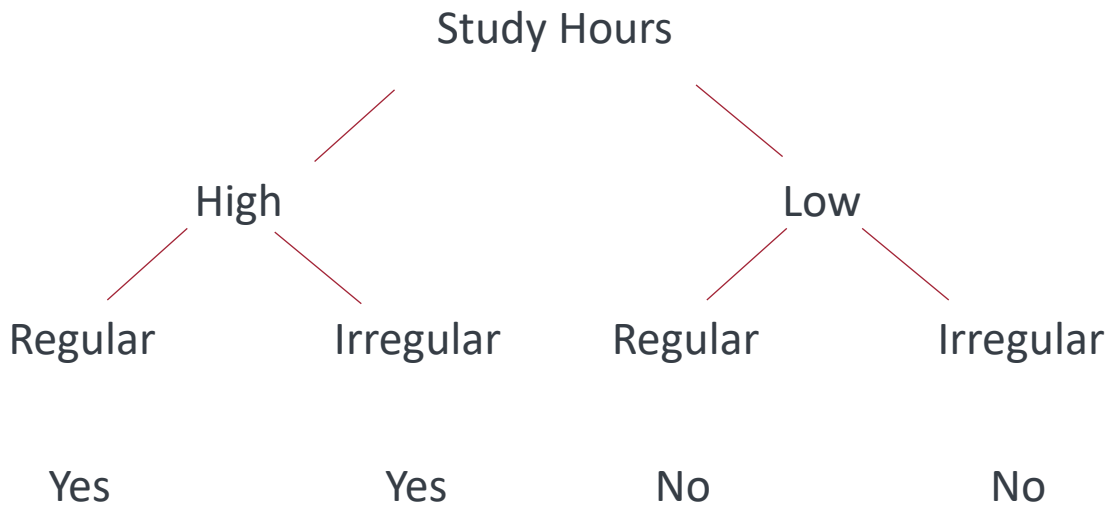
Let’s consider an example to classify whether a student passes an exam or not based on two features: study hours and attendance

Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes

The Decision Tree

Let’s consider an example to classify whether a student passes an exam or not based on two features: study hours and attendance

Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes



The subtree involving Medium Study Hours was pruned earlier, which simplified the tree.

Pruning a Subtree

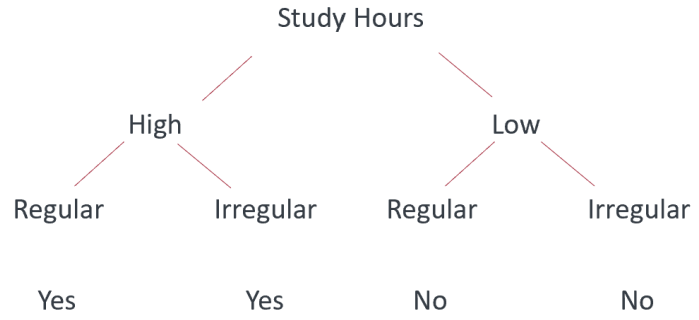
Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes

- Low Study Hours: we predict 'No' with regular and irregular attendance
- High Study Hours: we predict 'yes' with regular and irregular attendance
- At node 'Low study hours'
  - three students, B, F, and H,
  - among them, for H, we predicted incorrectly
  - so we ,have $e(t) = 1$ and $n(t) = 3$

$$err'_{T_t} = \frac{\Sigma(e(s) + 0.5)}{\Sigma n(s)} = \frac{(1 + 0.5) + (0 + 0.5)}{3} \approx 0.667$$

$$std(err'_{T_t}) = \sqrt{3 \cdot 0.667 \cdot (1 - 0.667)} = 0.816$$

$$e(t) \leq n_t \cdot err'_{T_t} + std(err'_{T_t})$$
$$1 \leq 3 \cdot 0.667 + 0.86 = 2.82$$



Since this condition holds, **we prune the subtree**

Do we also prune the subtree for 'High study hours'?

Chi-Squared pruning

- Chi-square measures the statistical significance of the differences between the child nodes and their parents
- It is computed by **the sum of squared standardized differences between observed and expected frequencies of target variable** for each node

$$\chi = \sqrt{\frac{(Actual - Expected)^2}{Expected}}$$

Chi-squared pruning: to test whether splitting on an attribute contributes a statistically significant amount of information.

-- Use the statistical significance test to find the irrelevant attribute.

Review of chi-squared test

If X_i are k independent, normally distributed random variables with mean 0 and variance 1, then the random variable is distributed according to the chi-square distribution:

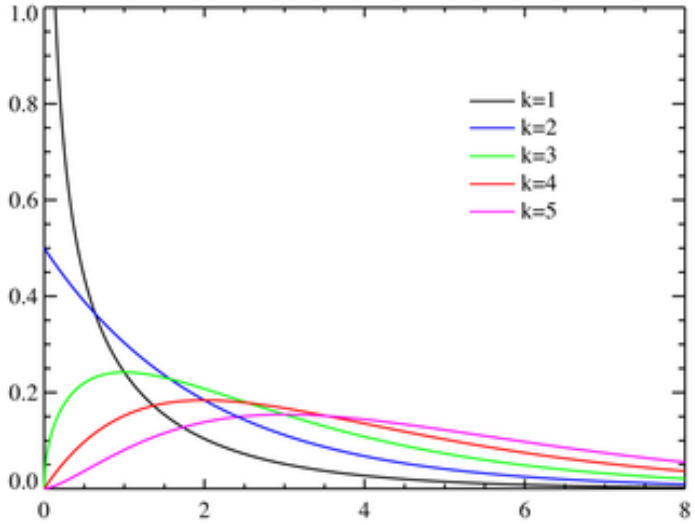
The chi-square distribution has one parameter: k - a positive integer that specifies the number of degrees of freedom.

A probability density function of the chi-square distribution is

$$f(x;k) = \begin{cases} \dfrac{1}{2^{k/2}\Gamma(k/2)}\, x^{k/2-1} e^{-x/2} & \text{for } x > 0, \\ 0 & \text{for } x \leq 0, \end{cases}$$

Where the Gamma function is:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, \mathrm{d}t$$



Review of chi-squared test

Suppose at one node in the tree have training data S, the number of positive examples in S is p, and the number of negative examples is n, then $p+n=|S|$

By the information gain analysis, we select an attribute that splits S into a number of subsets S_i , each of these subsets has p_i positive examples and n_i negative examples. (p_i + n_i = $|S_i|$)

If this attribute is irrelevant, then we would expect that the number of positive and negative examples in S_i would be:

$$\hat{p}_i = \frac{p}{p+n}|S_i| \qquad \hat{n}_i = \frac{n}{p+n}|S_i|$$

Then we can calculate how much "error" there is between the actual number of positive and negative examples in each S_i and the expected number.

$$Q = \sum_i \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

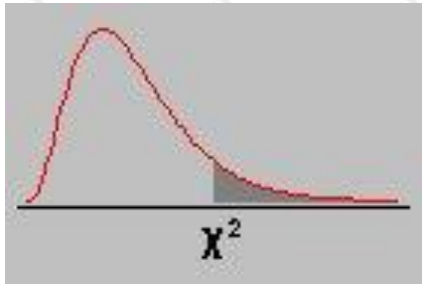
Note: this sum is over each subset created by the split on this attribute.

Review of chi-squared test

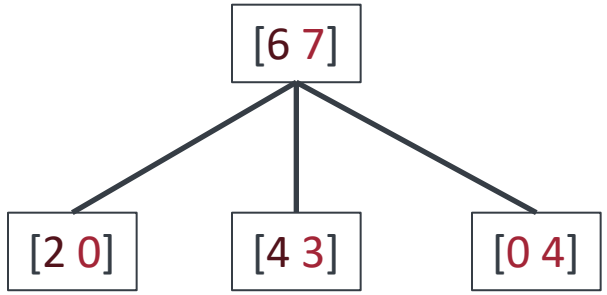
You can use a statistical software or Chi-Square Table to estimate the probability that the attribute is really irrelevant or not.

df\area	.995	.990	.975	.950	.900	.750	.500	.250	.100	.050	.025	.010	.005
1	0.00004	0.00016	0.00098	0.00393	0.01579	0.10153	0.45494	1.32330	2.70554	3.84146	5.02389	6.63490	7.87944
2	0.01003	0.02010	0.05064	0.10259	0.21072	0.57536	1.38629	2.77259	4.60517	5.99146	7.37776	9.21034	10.59663
3	0.07172	0.11483	0.21580	0.35185	0.58437	1.21253	2.36597	4.10834	6.25139	7.81473	9.34840	11.34487	12.83816
4	0.20699	0.29711	0.48442	0.71072	1.06362	1.92256	3.35669	5.38527	7.77944	9.48773	11.14329	13.27670	14.86026
5	0.41174	0.55430	0.83121	1.14548	1.61031	2.67460	4.35146	6.62568	9.23636	11.07050	12.83250	15.08627	16.74960
6	0.67573	0.87209	1.23734	1.63538	2.20413	3.45460	5.34812	7.84080	10.64464	12.59159	14.44938	16.81189	18.54758
7	0.98926	1.23904	1.68987	2.16735	2.83311	4.25485	6.34581	9.03715	12.01704	14.06714	16.01276	18.47531	20.27774
8	1.34441	1.64650	2.17973	2.73264	3.48954	5.07064	7.34412	10.21885	13.36157	15.50731	17.53455	20.09024	21.95495
9	1.73493	2.08790	2.70039	3.32511	4.16816	5.89883	8.34283	11.38875	14.68366	16.91898	19.02277	21.66599	23.58935
10	2.15586	2.55821	3.24697	3.94030	4.86518	6.73720	9.34182	12.54886	15.98718	18.30704	20.48318	23.20925	25.18818

What you need:
Q and degree of freedom (df)
Degrees of Freedom (df)=(Number of Classes−1)
×(Number of Branches−1)



Prune based on chi-squared test



p=6, n=7

MaxPchance: the worst chance we are willing to accept

$$\hat{p}_1 = \frac{6}{6+7} * 2 = 0.92 \qquad \hat{p}_2 = \frac{6}{6+7} * 7 = 3.23 \qquad \hat{p}_3 = \frac{6}{6+7} * 4 = 1.85$$

$$\hat{n}_1 = \frac{7}{6+7} * 2 = 1.08 \qquad \hat{n}_2 = \frac{7}{6+7} * 7 = 3.77 \qquad \hat{n}_3 = \frac{7}{6+7} * 4 = 2.15$$

$$Q = \sum_i \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

$$= \frac{(2-0.92)^2}{0.92} + \frac{(0-1.08)^2}{1.08} + \frac{(4-3.23)^2}{3.23} + \frac{(3-3.77)^2}{3.77} + \frac{(0-1.85)^2}{1.85} + \frac{(4-2.15)^2}{2.15}$$

$$= 6.13$$

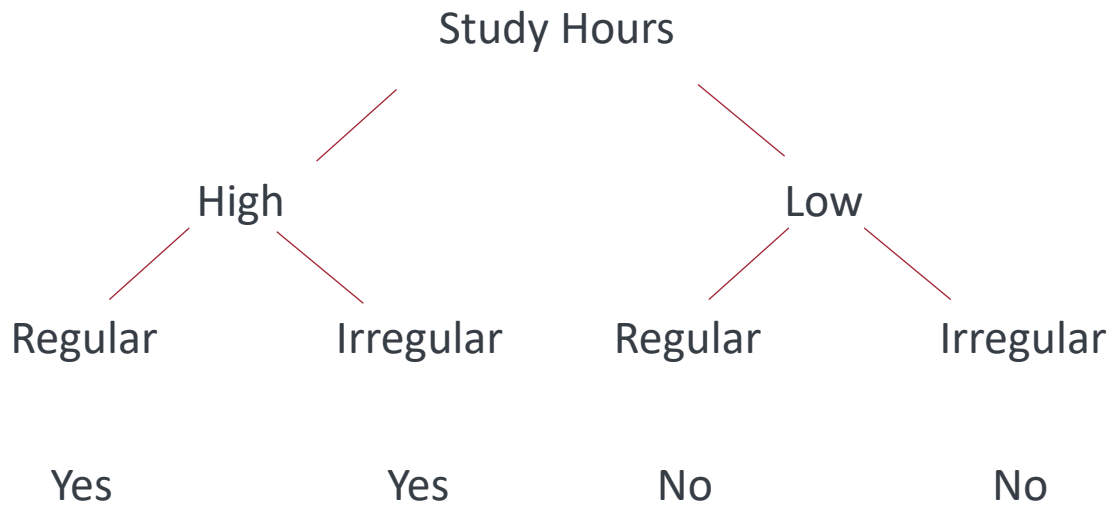
Compare this to a MaxPchance

Degree of freedom (df) = 3-1=2, so p_{chance} = 0.05, the critical value is 5.99, so no pruning

An Example

Let’s work on the same example

Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes

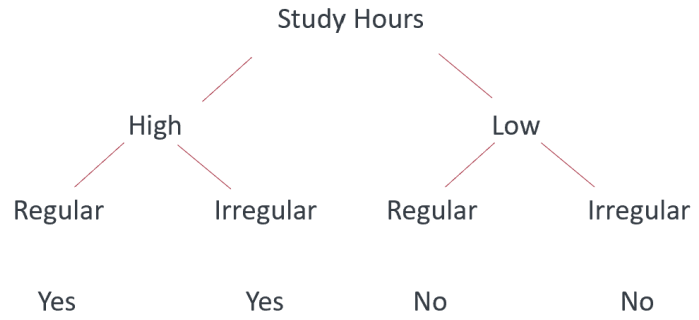


The subtree involving Medium Study Hours was pruned earlier, which simplified the tree.

Pruning a Subtree

- Let's focus on the subtree under the node "Low study hours"
- The branches split based on Attendance (regular or irregular), with the following class distributions for Pass Exam (Yes or No)
- Actual frequencies
  - Regular attendance: observed distribution- Pass: 1, Fail: 1
  - Irregular attendance: observed distribution- Pass: 0, Fail: 1
- Expected frequencies
  - Overall distribution: Pass: 1, Fail: 2 (total: 3), **regular attendance 2**, and irregular 1
  - Regular attendance: expected- Pass: (1/3)*__2__=0.667, Fail: (2/3)*__2__=1.33
  - Irregular attendance: expected- Pass: (1/3)*1=0.33, Fail: (2/3)*1=0.667

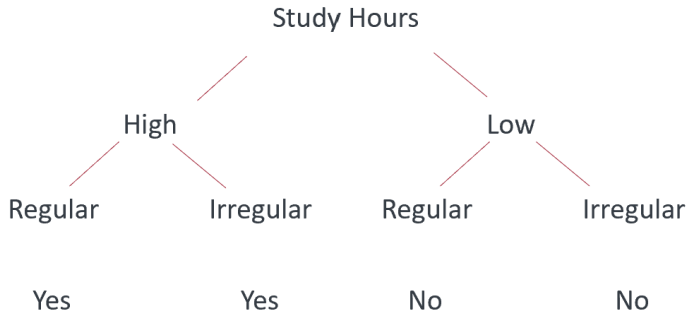
Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes



Pruning a Subtree

Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes

- Actual frequencies
  - Regular attendance: observed distribution- Pass: 1, Fail: 1
  - Irregular attendance: observed distribution- Pass: 0, Fail: 1
- Expected frequencies
  - Overall distribution: Pass: 1, Fail: 2 (total: 3), regular attendance 2, and irregular 1
  - Regular attendance: expected- Pass: (2/3)*1=0.667, Fail: (2/3)*2=1.33
  - Irregular attendance: expected- Pass: (1/3)*1=0.33, Fail: (1/3)*2=0.667
- Compute the Chi-squared Statistic:

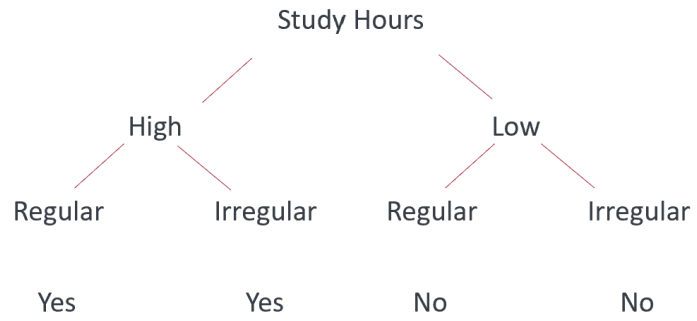


- Regular attendance: $\chi^2 = \sum \frac{(Actual - Expected)^2}{Expected} = \frac{(1 - 0.667)^2}{0.667} + \frac{(1 - 1.33)^2}{1.33} = 0.248$
- Irregular attendance: $\chi^2 = \sum \frac{(Actual - Expected)^2}{Expected} = \frac{(0 - 0.33)^2}{0.33} + \frac{(1 - 0.667)^2}{0.667} = 0.49$

Pruning a Subtree

Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes

- Compute the Chi-squared Statistic:
- Regular attendance: $\chi^2 = \sum \frac{(Actual - Expected)^2}{Expected} = 0.248$
- Irregular attendance: $\chi^2 = \sum \frac{(Actual - Expected)^2}{Expected} = 0.49$
- Total Chi-Squared Statistic: 0.248 + 0.49 = 0.73
- Compare with Critical Value:
  - Degrees of Freedom=(Number of Classes−1)×(Number of Branches−1)
    =(2-1)*(2-1)=1
  - The critical chi-squared value at a significance level of 0.05 and 1 degree of freedom is 3.841.



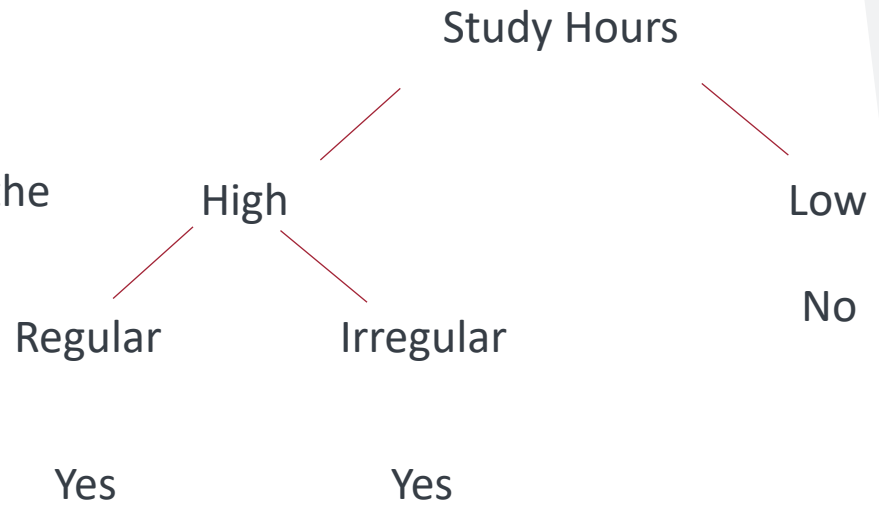
df\area	.995	.990	.975	.950	.900	.750	.500	.250	.100	.050	.025	.010	.005
1	0.00004	0.00016	0.00098	0.00393	0.01579	0.10153	0.45494	1.32330	2.70554	3.84146	5.02389	6.63490	7.87944

Pruning a Subtree

Student	Study Hours	Attendance	Pass Exam (Label)
A	High	Regular	Yes
B	Low	Regular	No
C	Medium	Irregular	No
D	Medium	Regular	Yes
E	High	Regular	Yes
F	Low	Irregular	No
G	High	Irregular	No
H	Low	Regular	Yes

- Total Chi-Squared Statistic: 0.248 + 0.49 = 0.73
- Compare with Critical Value:
  - Degrees of Freedom=(Number of Classes−1)×(Number of Branches−1)
    =(2-1)*(2-1)=1
  - The critical chi-squared value at a significance level of 0.05 and 1 degree of freedom is 3.841.
- 0.73 < 3.841, **so the split is not statistically significant.**
- The prediction for low study hours is “Fail”, which is the majority class in the branch.

Try this calculation with ‘High study hours’



Review of chi-squared test

Small χ^2 : this attribute is not relevant -- the data in each split are following the same distribution as the data before splitting on this attribute.

Large χ^2: means that there is a lot of "error" between what we would have expected (under the irrelevant attribute hypothesis) and the actual distribution of examples. The distribution (chi-squared distribution) will determine the probability that the attribute is not relevant.

Using Chi-squared to avoid overfitting

Build the full decision tree as before.

But when you can grow it no more, start to prune:

Beginning at the bottom of the tree, delete splits in which p_{chance} > *MaxPchance.*

Continue working you way up until there are no more prunable nodes.

MaxPchance is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise.

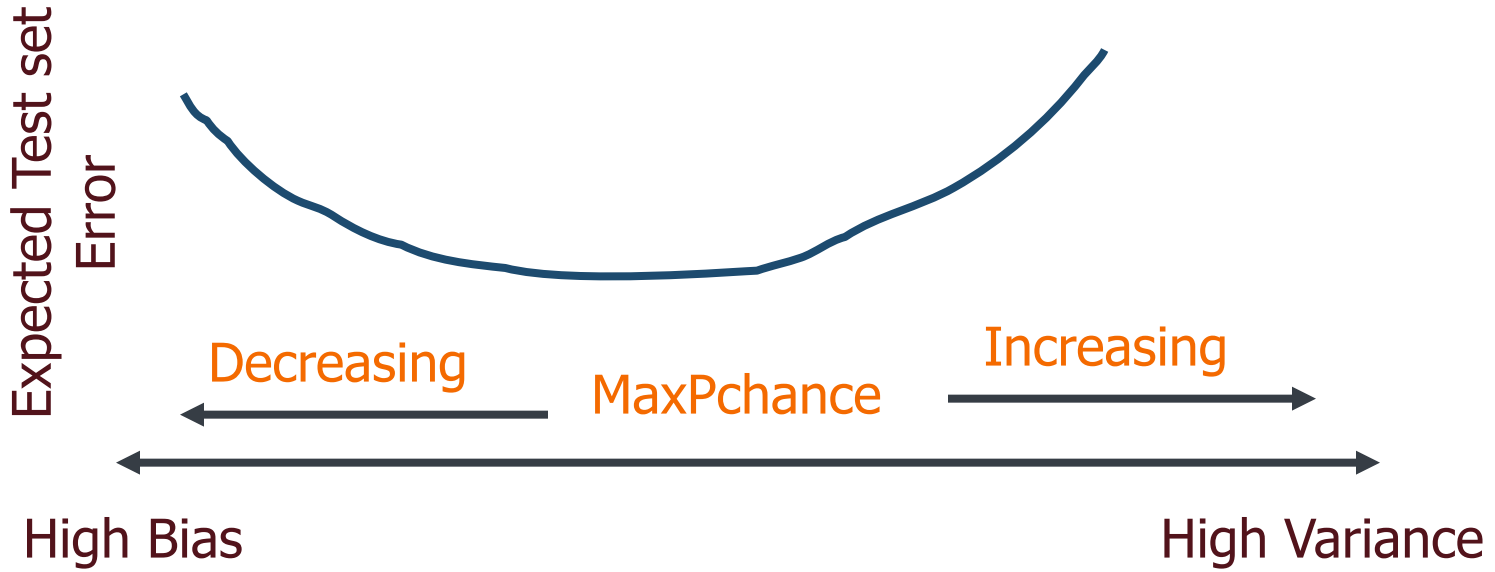
MaxPchance

Good news: The decision tree can automatically adjust its pruning decisions according to the amount of apparent noise and data.

Bad news: The user must come up with a good value of MaxPchance. (for instance, 0.05 as a magic parameter).

Good news: But with extra work, the best MaxPchance value can be estimated automatically by ***cross-validation***.

MaxPchance


Expected Test set Error
Decreasing
MaxPchance
Increasing
High Bias
High Variance

CART - Gini Impurity

- An alternative impurity measurement to entropy:

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

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i-th node.

- Scikit-Learn uses the Classification and Regression Tree (CART) algorithm to train Decision Trees. CART cost function:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

where G_{left} and G_{right} are gini impurity of left and right subset respectively; m_{left}/ m_{right} is the number of instances in the left/right subset. k, t_k are feature and its threshold respectively.

- Gini impurity vs. Entropy
  - Gini is slightly faster
  - Entropy tends to produce slightly more balanced trees
  - Most of the time they lead to similar trees

Training Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

Example

Compute the gini impurity for attribute 'Outlook'

	Target = Yes	Target = No	Total
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

$$gini('Outlook = Sunny') = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$gini('Outlook = Overcast) = 1 - \left(\frac{4}{4}\right)^2 - (0)^2 = 0$$

$$gini('Outlook = Rain) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$gini('Outlook) = \frac{5}{14} \cdot 0.48 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.48 = 0.342$$

Overall Gini Impurity = 0.459

$gini('Outlook)$**= 0.342**

Similarly, you can do this for all other attributes, choose the attribute with smallest gini impurity to split.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK **YOU**

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030