Part-2: Decision Tree Classifier

Training and Testing

Training and Testing

	Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Training set	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
Validation	11	Sunny	Mild	Normal	Strong	Yes
	12	Overcast	Mild	High	Strong	Yes
Test	13	Overcast	Hot	Normal	Weak	??
	14	Rain	Mild	High	Strong	

Training Set

- It's the set of data used to train the model.
- During each epoch, our model will be trained over and over again on this same data in our training set, and it will continue to learn about the features of this data.

Training set

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	
14	Rain	Mild	High	Strong	

Validation Set

- To validate our model during training.
- This validation process gives information about adjusting hyperparameters (ex. learning rate).
- Validation set is to ensure that our model is not overfitting to the data in the training set.

- **The validation set allows us to see how well the model is generalizing during training.**

Validation

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	
14	Rain	Mild	High	Strong	

Test Set

The test set provides a final check that the model is generalizing well before deploying the model to production.

- The test set is a set of data that is used to test the model after the model has already been trained.
- The test set is separate from both the training set and validation set.

- After our model has been **trained** and **validated** using our training and validation sets, we will then use our model to **predict the output of the unlabeled data** in the test set.

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	
14	Rain	Mild	High	Strong	

Test

Difference between the test set and the two other sets is that the **test set should not be labeled**. The **training set and validation set have to be labeled.**

K-fold Cross validation technique

- Cross-validation is a statistical method used to estimate the ability of machine learning models.

- It is commonly used to compare and select a model.

- The parameter k refers to the number of groups that a given data sample is to be split into.

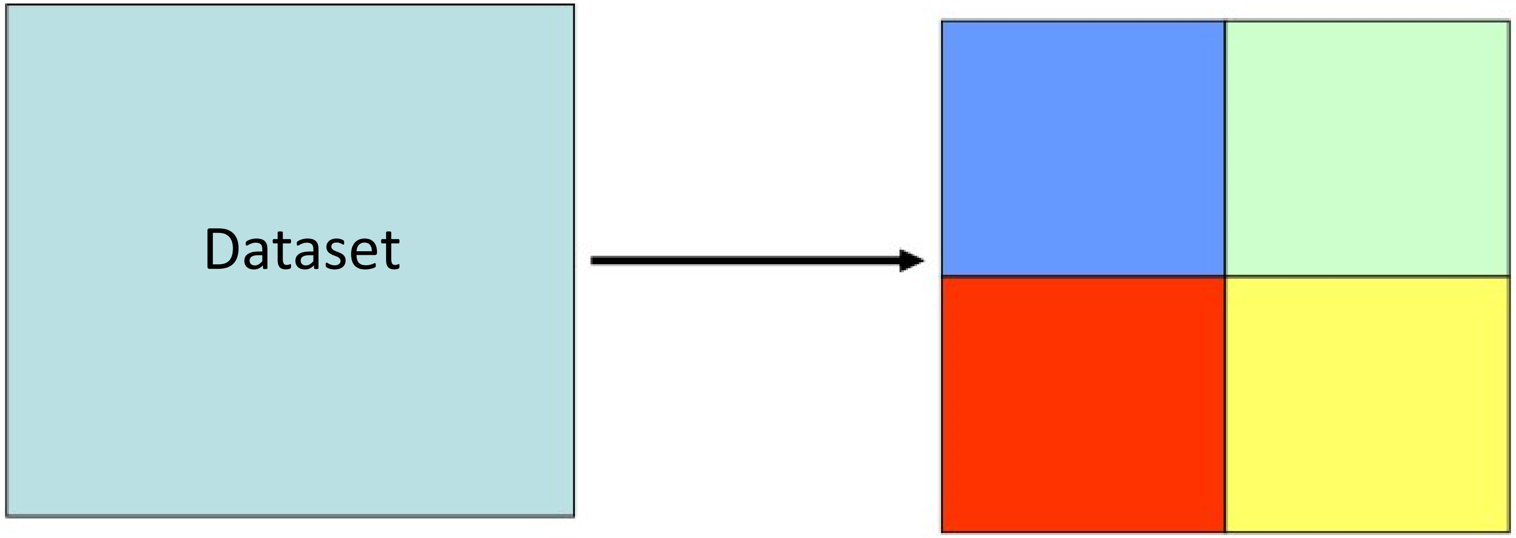
- The procedure is often called k-fold cross-validation.

- When k=10 becoming 10-fold cross-validation

General procedure

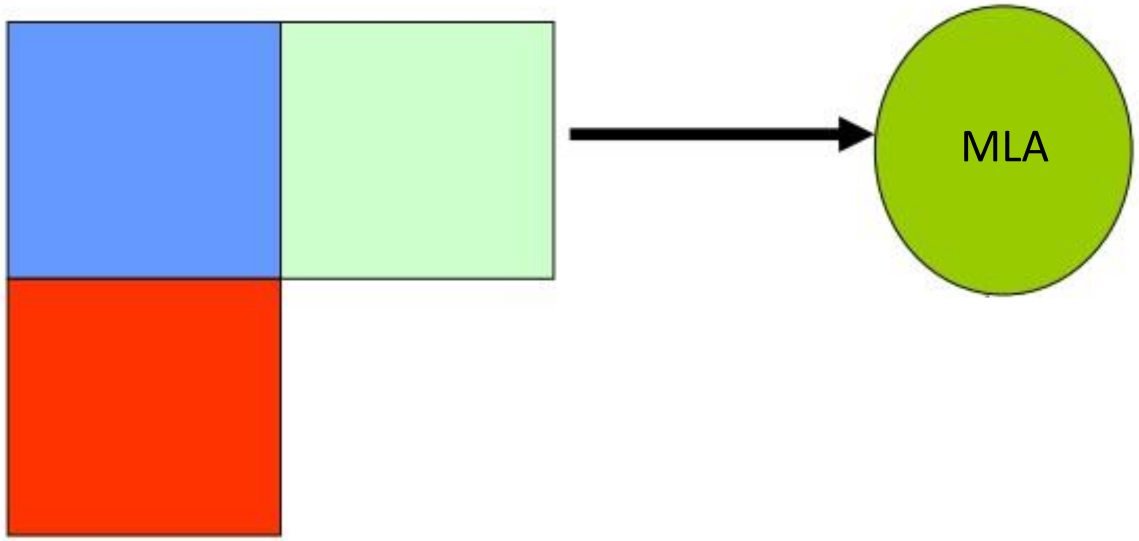
1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

4-Fold Cross-validation



Dataset is partitioned randomly into 4 equal sets

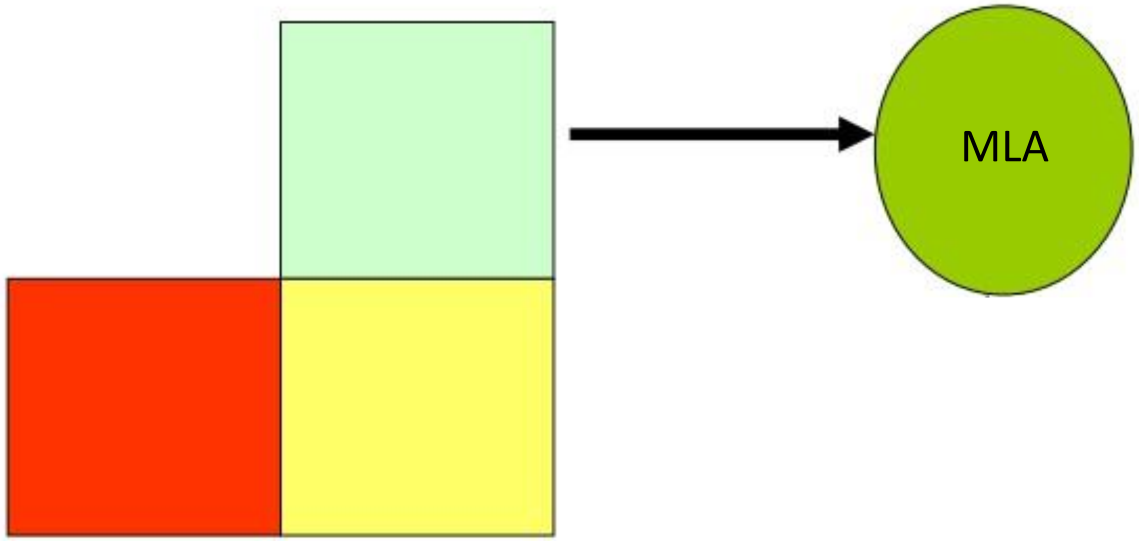
4-Fold Cross-validation



Training Dataset

Training and validation of each classifier were carried out 4 times using one distinct set for testing and other 4-1 sets for training.

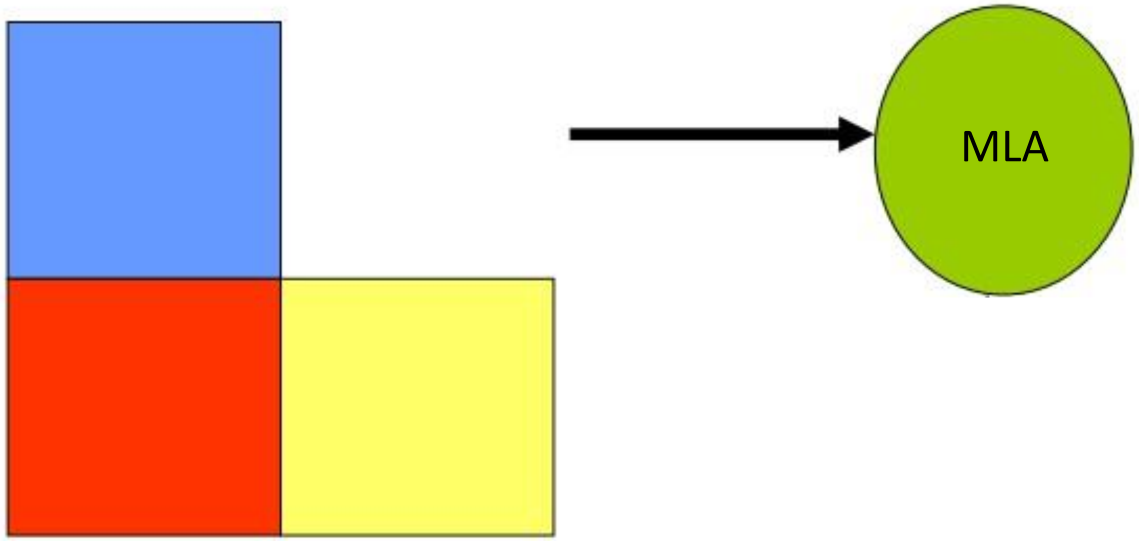
4-Fold Cross-validation



Training Dataset

Training and validation of each classifier were carried out 4 times using one distinct set for testing and other 4-1 sets for training.

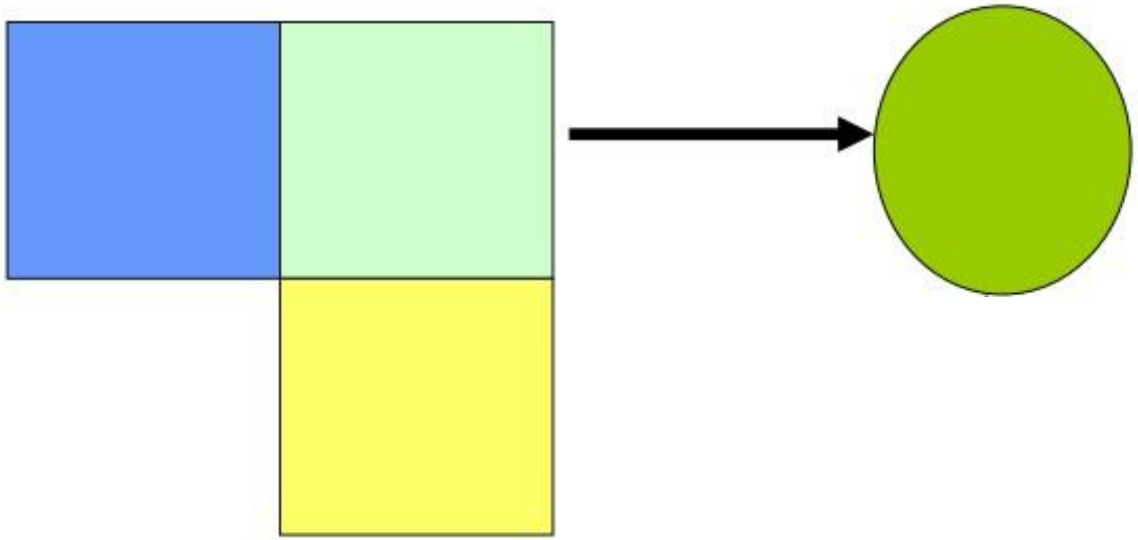
4-Fold Cross-validation



Training Dataset

Training and validation of each classifier were carried out 4 times using one distinct set for testing and other 4-1 sets for training.

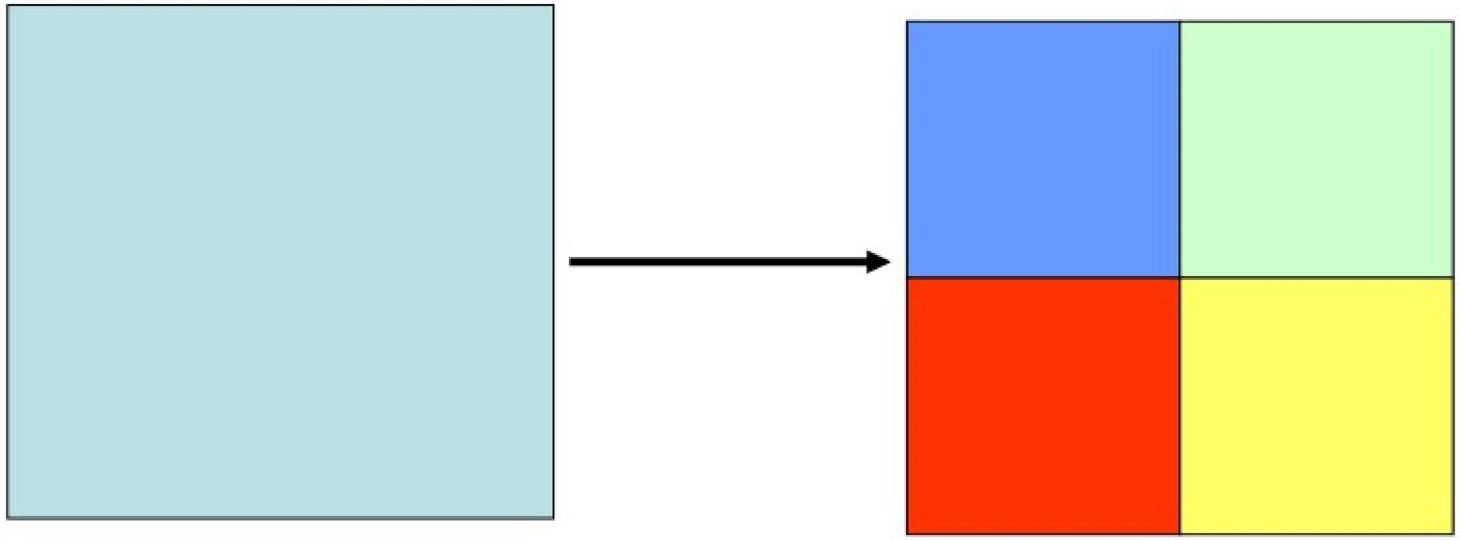
4-Fold Cross-validation



Training Dataset

Training and validation of each classifier were carried out 4 times using one distinct set for testing and other 4-1 sets for training.

4-Fold Cross-validation



ACC = (ACC1 + ACC2 + ACC3 + ACC4) / 4

Training and validation of each classifier were carried out 4 times using one distinct set for testing and other 4-1 sets for training.

Selection or Configuration of k

- The k value must be chosen carefully for your data sample.

- A poorly chosen value for k may result in a misrepresentative idea of the skill of the model, such as a score with a **high variance** (that may change a lot based on the data used to fit the model), or a **high bias**, (such as an overestimate of the skill of the model).

- Three common tactics for choosing a value for k are as follows:
  - **Representative**
  - **K=5 or 10**
  - **K=n**

- **Representative**:
  - The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.

- **k=10**:
  - The value for k is fixed to 10, a value that has been found through experimentation to generally result in a model skill estimate with low bias & a modest variance.

- **k=n**:
  - The value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called **leave-one-out cross-validation (LOOCV)**.

Cross-Validation API

The **scikit-learn library** provides an implementation that will split a given data sample up

KFold (*number of splits*, *whether or not to shuffle the sample*, seed for the pseudorandom number generator used prior to the shuffle **)** scikit-learn class can be used.

KFold (3 folds, shuffles prior to the split, uses a value of 1 for the pseudorandom number generator).

```
kfold = KFold(3, True, 1)

# split function
for train, test in kfold.split(data):
	print('train: %s, test: %s' % (train, test))
```

Issues in Decision Tree Learning

- Determining how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs, and improving computational efficiency

(1) Determining how deeply to grow the decision tree

Overfitting

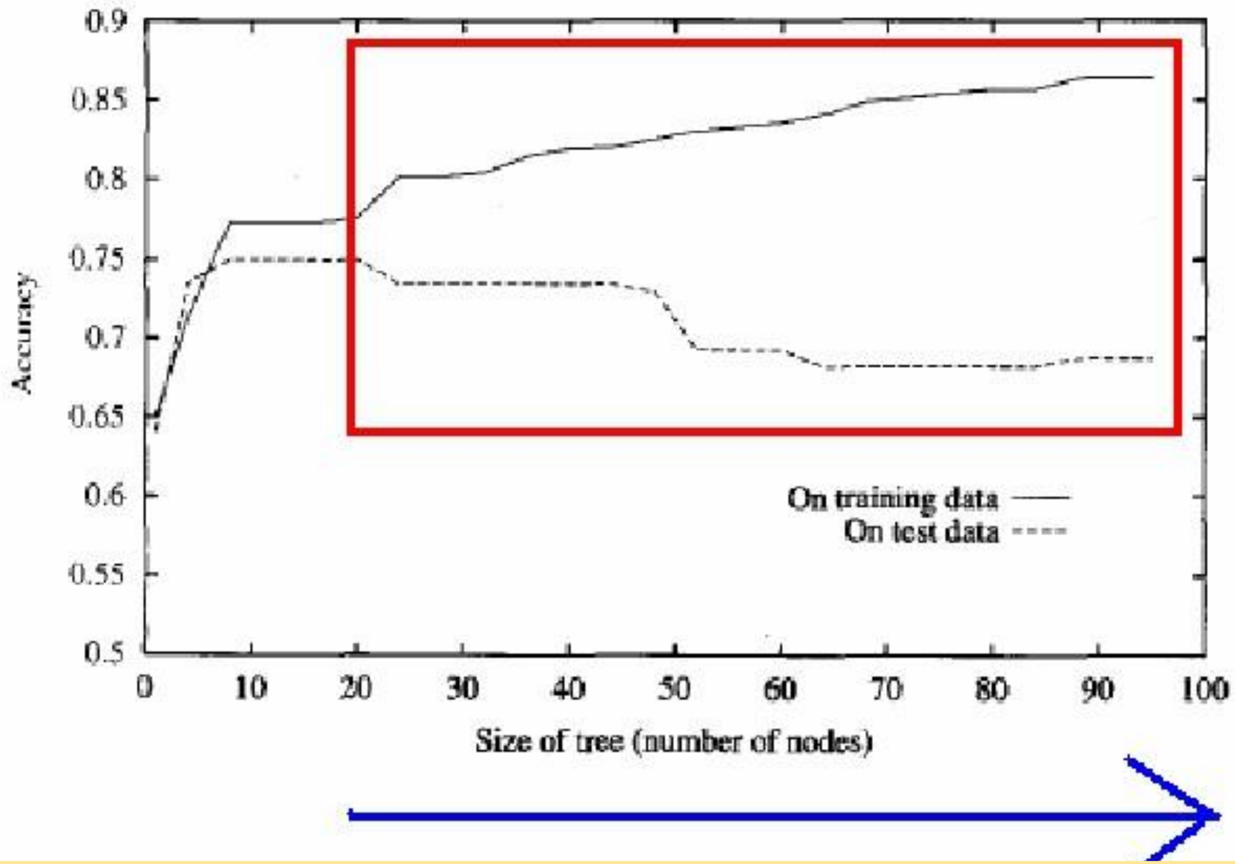
- Training: Task of fitting the model/ algorithm to a set of training data, in order to make the reliable prediction on unseen test data

- It is difficult to produce a representative sample of the true target function
  - when there is noise in the data or
  - when the number of training examples is too small.

- In either of these cases, ID3 algorithm can produce trees that ***overfit*** the training examples.

(1) Determining how deeply to grow the decision tree

Overfitting in Decision Tree



Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases.

(1) Determining how deeply to grow the decision tree

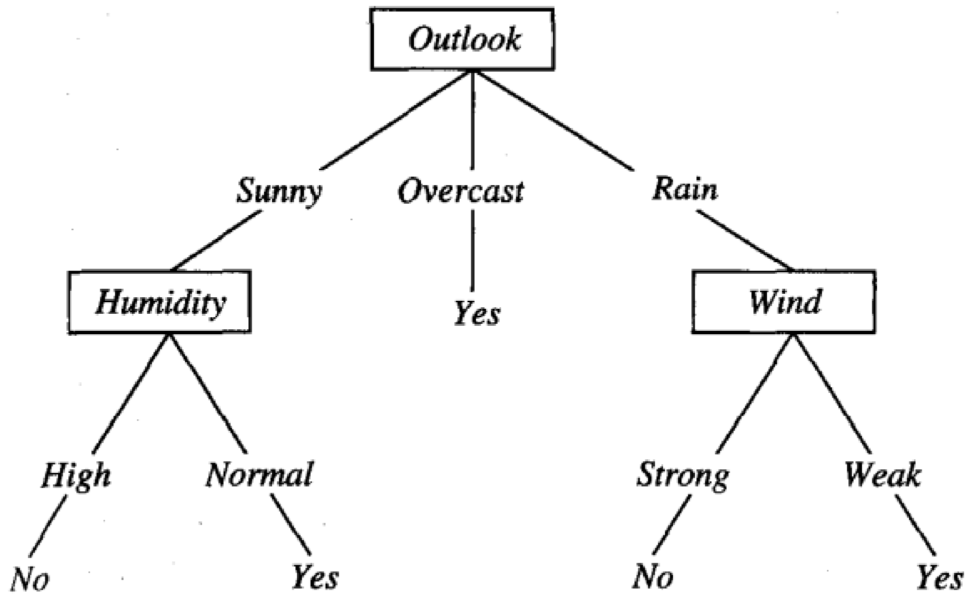
Why Overfitting Happens in Decision Tree Learning?

- **Presence of error in the training examples. (common in machine learning)**
- **When small numbers of examples are associated with leaf nodes.**

(1) Determining how deeply to grow the decision tree

Presence of Error and Over-fitting

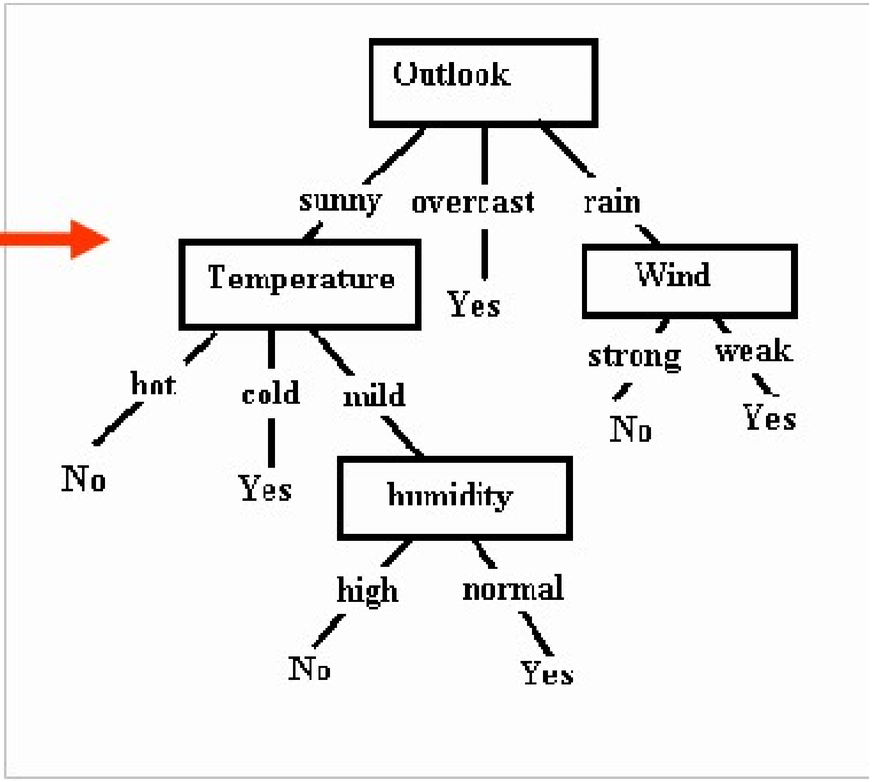
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool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



(D15 Outlook = Sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = Yes)

Presence of Error and Over-fitting

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

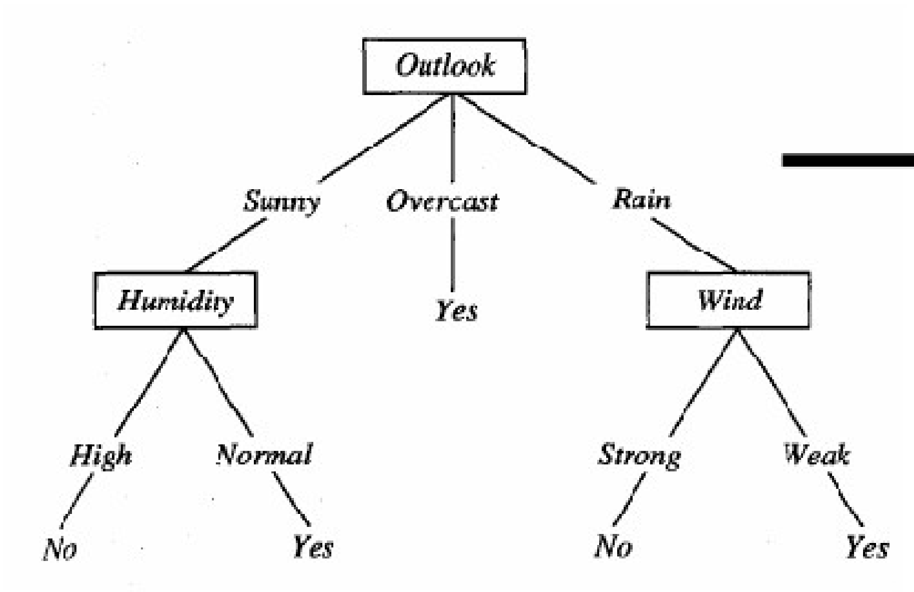


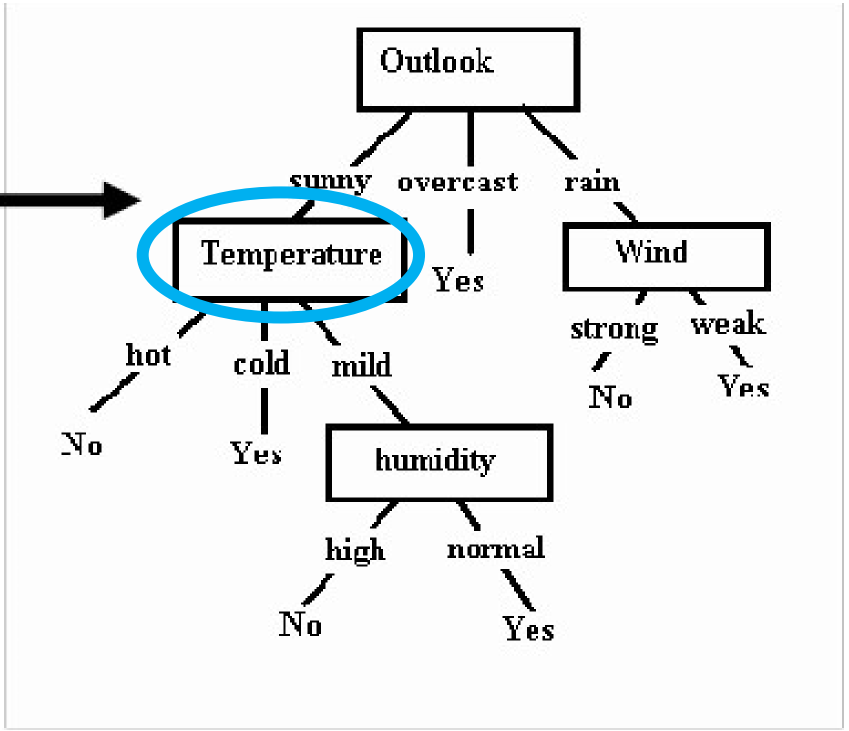
+

$$\langle Outlook = Sunny, Temperature = Hot, Humidity = Normal, \quad Wind = Strong, PlayTennis = No\rangle$$

This example illustrates how random noise in the training examples can lead to overfitting.

Presence of Error and Over-fitting





Whether Tom will play tennis or not on D16?

	OL	Tem	Hum	Wind
D16	Sunny	Mild	High	Weak ?

More Complex
Tree depth is more

Why to avoid Overfitting ??

Experimental study of ID3 with noisy data → overfitting

Decrease the accuracy of decision trees by **10-25%** on most problems.

How to avoid Overfitting

There are several approaches to avoiding overfitting in decision tree learning. These can be grouped into two classes:

- Approach-1: Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
  - Direct approach
  - Difficult to estimate precisely **when to stop** growing the tree.

- Approach-2: Allow the tree to overfit the data, and then post-prune the tree – found to be more successful in practice

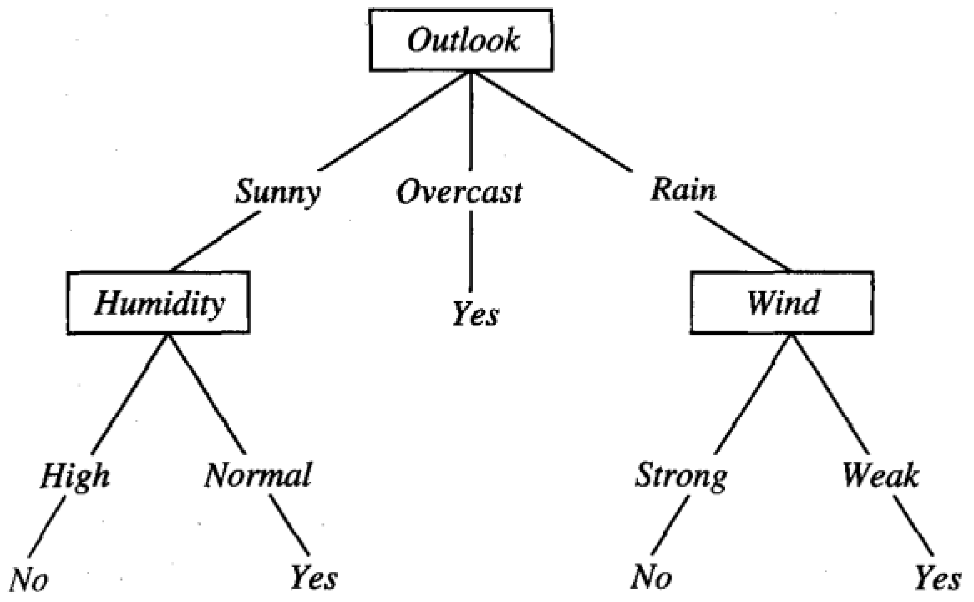
Pruning Methods

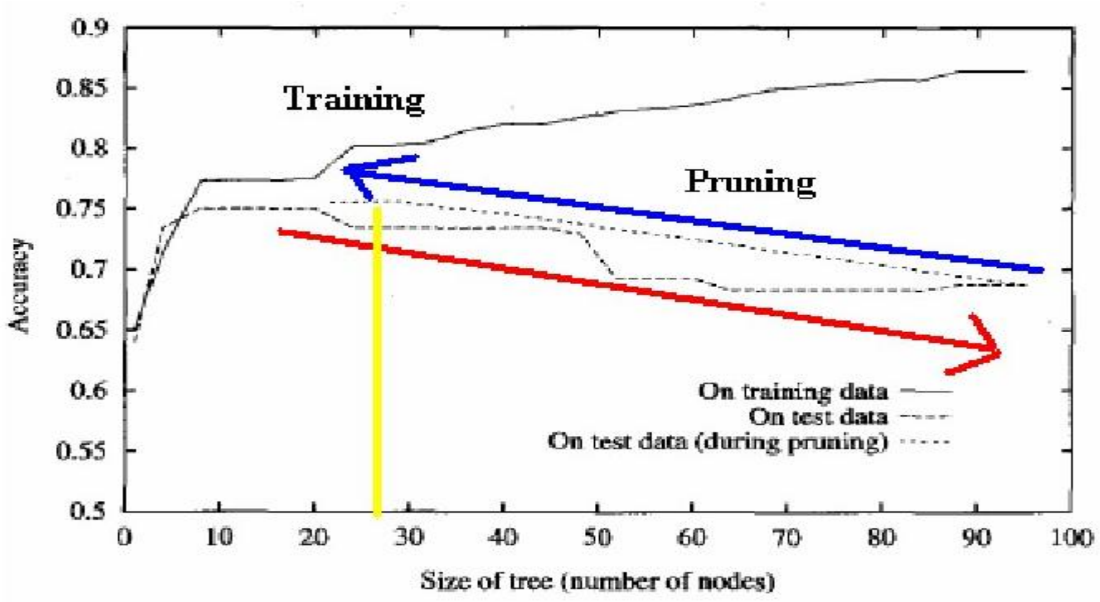
- Reduced-error pruning (Quinlan 1987)
- Rule post-pruning (Quinlan 1993)

Reduced Error Pruning

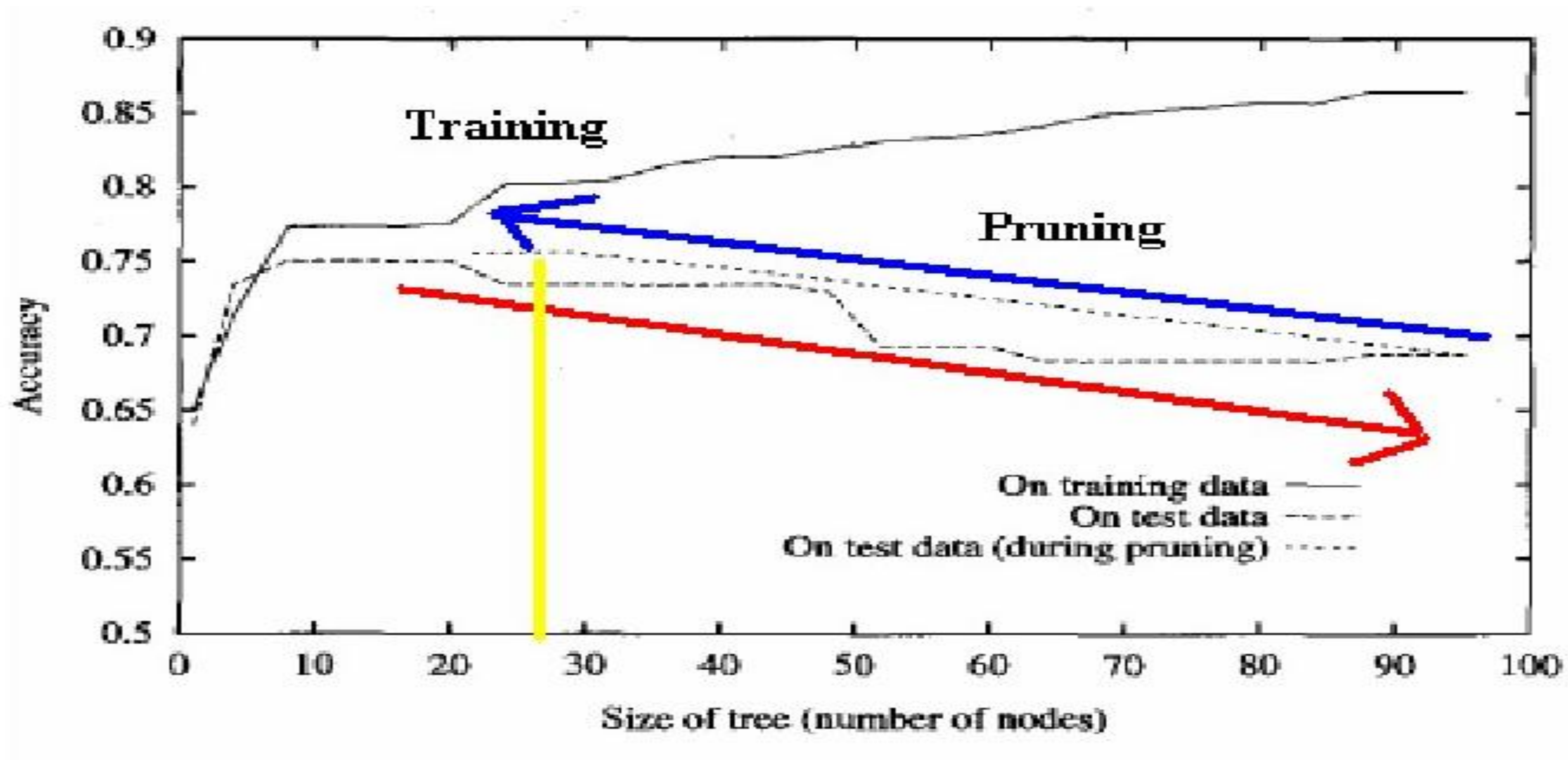
- Pruning a decision node:
  - Removing the subtree rooted at that node
  - Make it a leaf node
  - Assigning it the most common classification of the training examples affiliated with that node.

- Nodes are removed only if the resulting pruned tree performs no worse than-the original over the validation set.





Reduced Error Pruning



Rule Post-Pruning

In practice, it is one quite successful method for finding high accuracy hypotheses in post-pruning of decision tree.

Rule Post-Pruning

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.

2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.

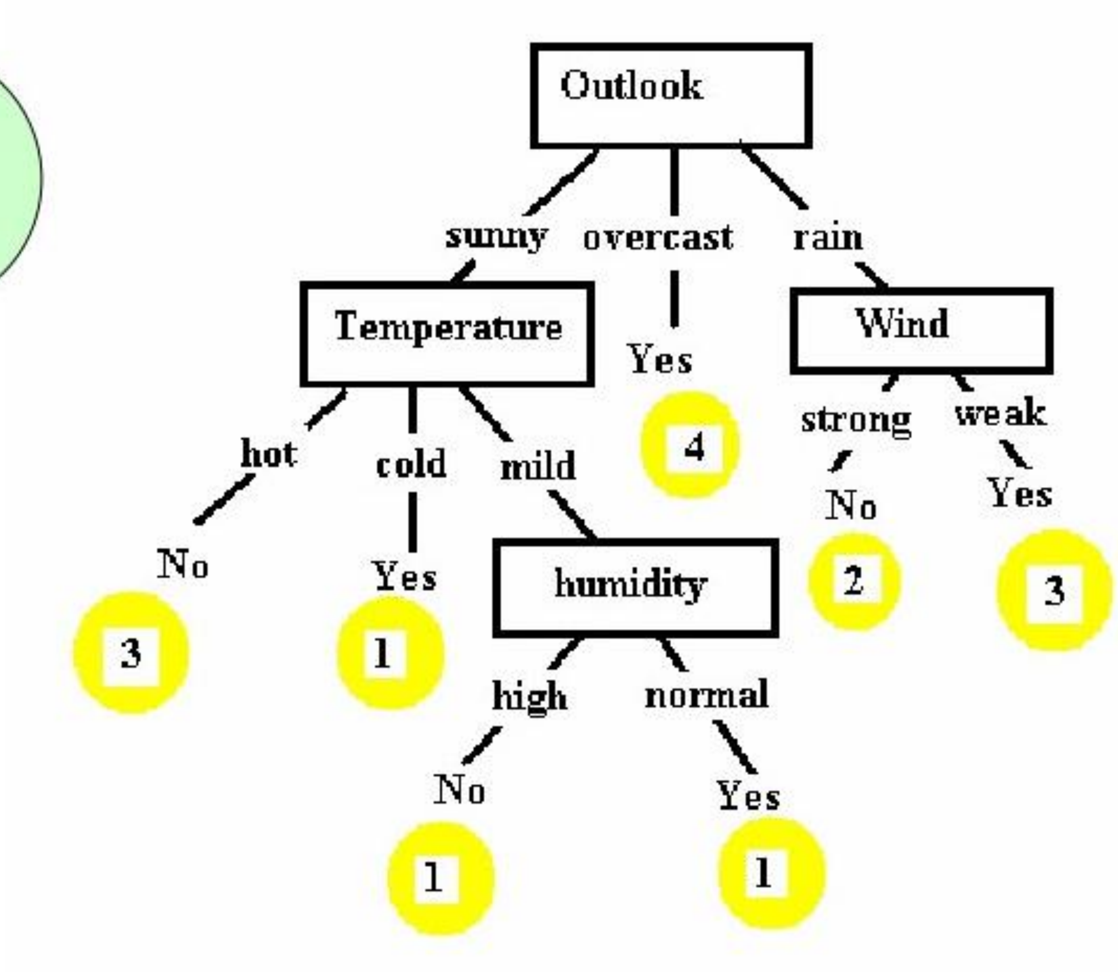
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.

4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Rule Post-Pruning (Step 1)

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.

1



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
D15	Sunny	Hot	Normal	Strong	No

Rule Post-Pruning (Step 2)

2

Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.

1: IF (Outlook = sunny and Temperature = Hot) **THEN** PlayTennis = No

2: IF (Outlook = sunny and Temperature = Cold) **THEN** PlayTennis = Yes

3: IF (Outlook = sunny and Temperature = Mild and Humidity=High) **THEN** PlayTennis = No

4: IF (Outlook = sunny and Temperature = Mild and Humidity=Normal) **THEN** PlayTennis = Yes

5: IF (Outlook = overcast) **THEN** PlayTennis = Yes

6: IF (Outlook = rain and Wind = Strong) **THEN** PlayTennis = No

7: IF (Outlook = rain and Wind = Weak) **THEN** PlayTennis = Yes

Rule Post-Pruning (Step 3)

3

Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.

1: IF (Outlook = sunny and Temperature = Hot) **THEN** PlayTennis = No

IF (Outlook = sunny and Temperature = Hot) **THEN** PlayTennis = No

IF (Outlook = sunny) **THEN** PlayTennis = No

IF (Temperature = Hot) **THEN** PlayTennis = No

(Validation examples)

Rule Post-Pruning (Step 3)

3

Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.

IF (Outlook = sunny and Temperature = Hot) **THEN** PlayTennis = No

Acc1 = 85%

IF (Outlook = sunny) **THEN** PlayTennis = No

Acc2 = 89%

IF (Temperature = Hot) **THEN** PlayTennis = No

Acc3 = 77%

(Validation examples)

If Acc2 > Acc3 & Acc1

1: IF (Outlook = sunny and Temperature = Hot) **THEN** PlayTennis = No

IF (Outlook = sunny) THEN PlayTennis = No

Rule Post-Pruning (Step 4)

4

Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

R1: Acc4
R2: Acc3
R3: Acc2
R4: Acc1

R11: Acc14
R12: Acc13
R13: Acc12
R14: Acc11

Sort rules in descending order of their accuracy on test dataset or validation examples

S1: Acc1
S2: Acc2
S3: Acc3
S4: Acc4

S11: Acc11
S12: Acc12
S13: Acc13
S14: Acc14

S1: Acc1 >= S2: Acc2 >= S3: Acc3 >= S4: Acc4 >= … >= S11: Acc11 >= S12: Acc12 >= S13: Acc13 >= S14: Acc14

Issues in Decision Tree Learning

- Determining how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs, and improving computational efficiency

Handling Continuous-Valued Attribute

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

Handling Continuous-Valued Attribute

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

Continuous values

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Handling Continuous-Valued Attribute

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

We have dynamically defined new discrete valued attributes so that it partition the continuous attribute value into a discrete set of intervals.

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Issues in Decision Tree Learning

- Determining how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs, and improving computational efficiency

Alternative Measures for Selecting Attributes

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

- We use information gain measure to select attributes as root
- Consider the attribute Day, which has a very large number of possible values.
- If it would be selected as the decision attribute for the root node of the tree and lead to a (quite broad) tree of depth one, which perfectly classifies the training data.
- However, this decision tree would classify poorly on subsequent examples, because it is not a useful predictor – overfit.

Alternative Measures for Selecting Attributes

- One way to avoid this difficulty is to select decision attributes based on some measure other than information gain.
- One alternative measure that has been used successfully is the gain ratio (Quinlan 1986).
- The gain ratio measure uses split information, that is sensitive to how broadly and uniformly the attribute splits the data.

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$

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

Gain Ratio (S, OL) = **0.1559**

Gain Ratio (S, Temp) = 0.01863

Gain Ratio (S, Hum) = 0.1475

Gain Ratio (S, Wind) = 0.0487

Find the Gain Ratio (S, Day) and decide the root element.

Issues in Decision Tree Learning

- Determining how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs, and improving computational efficiency

Mean
Average
Most repeated/ occurred

Issues in Decision Tree Learning

- Determining how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs, and improving computational efficiency

Handling Attributes with Different Cost

- In some learning tasks, the instance attributes may have associated costs.
- For example, in learning to classify medical diseases
  - we might describe patients in terms of attributes such as Temperature, BiopsyResult, Pulse, BloodTestResults, etc.
- These attributes vary significantly in their costs, both in terms of monetary cost and cost to patient comfort.
- In such tasks, we would prefer decision trees that use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classifications.

$$\textbf{select attribute based on cost} \rightarrow \frac{\textbf{Gain}(\textbf{S},\textbf{A})}{\textbf{Cost}(\textbf{A})}$$

Summary of issues in Decision Tree Learning

- Determining how deeply to grow the decision tree – pruning methods
- Handling continuous attributes - defining new discrete valued attributes
- Choosing an appropriate attribute selection measure – Gain ratio
- Handling training data with missing attribute values – most common
- Handling attributes with differing costs – consider cost factor of the attribute

improving computational efficiency

Decision Tree Approach-2
Measure of impurity: Gini Index

Gini Index

Gini index or Gini impurity measures the **degree or probability** of a particular variable being wrongly classified when it is randomly chosen.

- It means an attribute with lower Gini index should be preferred.

- The degree of Gini index varies between **0 and 1**.
  - **0 denotes** that all elements belong to a certain class or if there exists only **one class**
  - **1 denotes** that the elements are randomly **distributed across various classes**.

- A Gini Index of **0.5** denotes **equally distributed** elements **into some classes**.

Gini Index

- The Formula for the calculation of the of the Gini Index is given below.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

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

The most notable types of decision tree algorithms are:-

1. **Iterative Dichotomiser 3 (ID3):** This algorithm uses **Information Gain** to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.

2. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either **Information gain or Gain ratio** to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can **handle both continuous and missing attribute values**.

3. **Classification and Regression Tree (CART):** It is a dynamic learning algorithm which can produce a **regression tree** as well as a **classification tree** depending upon the **dependent variable**.

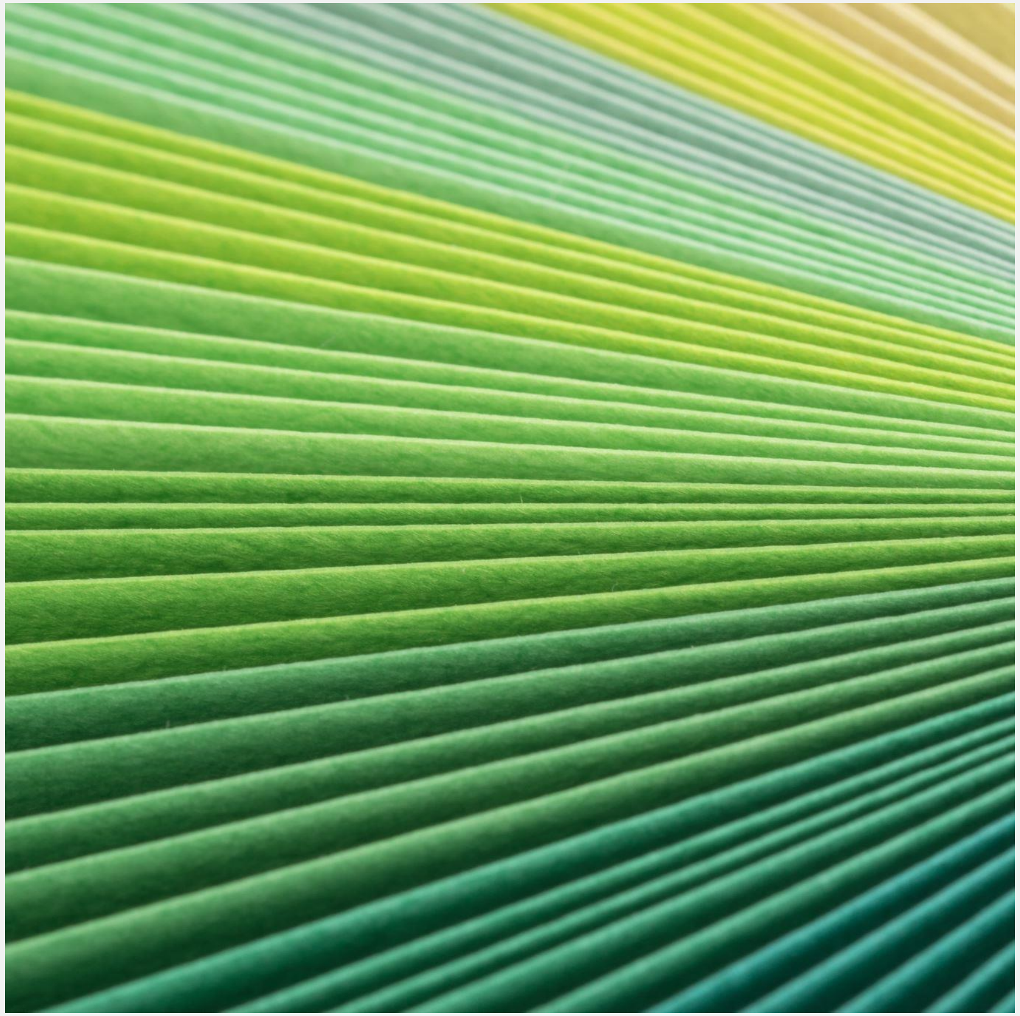
Decision Tree Regressor

Decision Tree - Regressor

outlook	temperature	humidity	wind	hours payed
Rain	hot	high	FALSE	25
Rain	hot	high	TRUE	30
Overcast	hot	high	FALSE	48
sunny	mild	high	FALSE	45
sunny	cool	normal	FALSE	52
sunny	cool	normal	TRUE	23
Overcast	cool	normal	TRUE	43
Rain	mild	high	FALSE	35
Rain	cool	normal	FALSE	38
sunny	mild	normal	FALSE	48
Rain	mild	normal	TRUE	48
Overcast	mild	high	TRUE	52
Overcast	hot	normal	TRUE	44
sunny	mild	high	FALSE	30

Summary

- What is DT
- Uses
- Types
- DT based on information gain
- DT based on Gini
- Issues in DT and methods to address them
- Training-test-validation
- Overfitting in DT



Thank you