*Concept Learning:* Task of a ML system to learn categories or concepts from training data.

*General Hypothesis (G):* A broad hypothesis that includes all possible instances, represented by unspecified attributes ('?' placeholders).

*Specific Hypothesis (S):* A detailed hypothesis specifying exact attributes ('pi' values) based on the training data.

*Version Space:* The set of all hypotheses that are consistent with the training data, lying between the general and specific hypotheses.

**Concept Space**

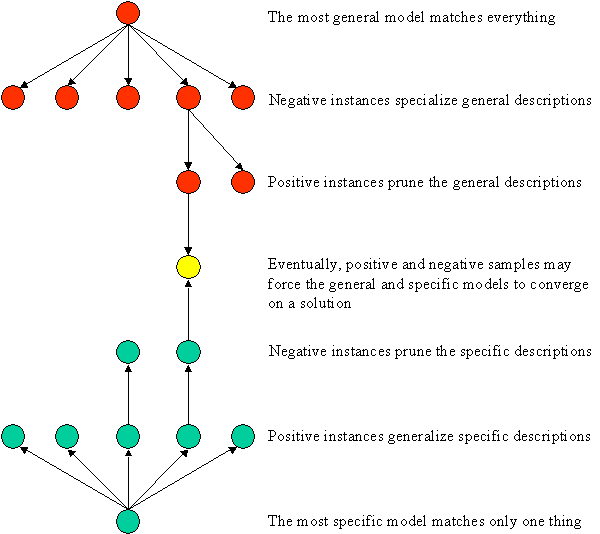
Objects: Represented as obj(X, Y, Z). Examples:

obj(X, Y, ball) obj(X, red, Z) obj(small, Y, Z) obj(X, red, ball)

obj(small, Y, ball) obj(small, red, Z) obj(small, red, ball) obj(small, orange, ball)

**Learning in Version Space: Generalization Operators**

1. *Replace constants with variables:* color(ball, red) → color(X, red)
2. *Remove literals from conjunctions:* shape(X, round) ∧ size(X, small) ∧ color(X, red) → shape(X, round) ∧ color(X, red)
3. *Add disjunctions:* shape(X, round) ∧ size(X, small) ∧ color(X, red) → shape(X, round) ∧ size(X, small) ∧ (color(X, red) ∨ color(X, blue))
4. *Replace a class with superclass in is-a relations:* is-a(tom, cat) → is-a(tom, animal)



**Version Space:** Set of all concept descriptions

consistent with learning/training examples.

Goal: Reduce the version space based on

learning examples.

Algorithms:-

1. From Specific to General
2. From General to Specific
3. Bidirectional Search: Eliminating those

inconsistent with training data, from both most

specific & most general hypotheses.

**Generalization** involves making a hypothesis

broader to include a wider range of instances.

**Specialization** involves making a hypothesis

more specific to exclude certain instances.

Cover all positive examples and specific enough

to exclude all negative examples.

*Overgeneralization:* If concept is too general obj(X, Y, Z)), it may cover +ve & -ve instances.

To avoid Overgeneralization: Generalize minimally to cover +ve examples. Use -ve instances to eliminate overly general concepts by specializing them.

**Algorithm for Searching from Specific to General**

1. *Initialize:*

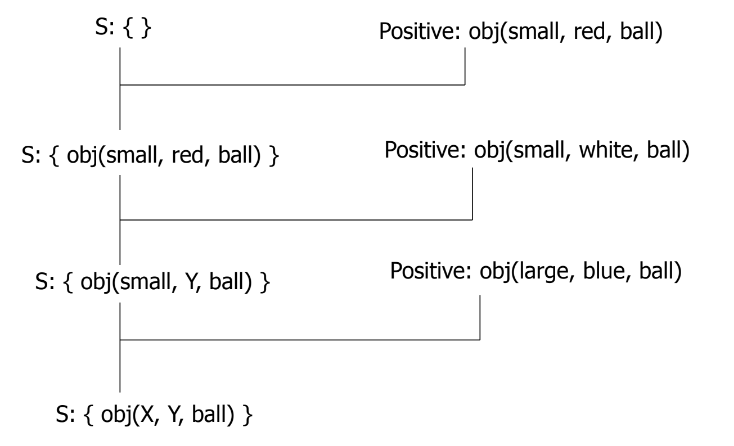
S: Set of specific hypotheses, starting with the first positive example.

N: Set of negative examples, initially empty.

1. *For each learning example:*
2. If positive (ex+, p):

For each hypothesis s∈S: If s does not cover p, replace s with the most specific generalization that covers p. Remove hypotheses in S that are more general than others or that cover any negative examples in N.

1. If negative (ex-, n):

Remove hypotheses in S that cover n. Add n to N to check for overgeneralization.

***Example:***

Initial S: {}

*First positive:* obj(small, red, ball)

Update S: Since S is empty, initialize S with the first positive example. {obj(small, red, ball)}

*Next positive:* obj(small, white, ball)

Update S: Current hypothesis in S (obj(small, red, ball)) doesn’t cover obj(small, white, ball). Generalize obj(small, red, ball) to obj(small, Y, ball), where Y can be any color.

S: { obj(small, Y, ball) }

*Next positive:* obj(large, blue, ball)

Update S: The current hypothesis in S (obj(small, Y, ball)) doesn’t cover obj(large, blue, ball). Generalize obj(small, Y, ball) to obj(X, Y, ball), where X can be any size.

S: { obj(X, Y, ball) }

**Algorithm for Searching from General to Specific**

1. *Initialize:*

G: Set of general hypotheses, starting with the most general description.

P: Set of positive examples, initially empty.

1. *For each learning example:*
2. If negative (ex-, n):

For each hypothesis g∈G: If g covers n, replace g with the most general specialization that does not cover n.

1. If positive (ex+, p):

Remove hypotheses in G that don’t cover p. Add p to P to check for overspecializa

***Example:***

Initial G: {obj(X, Y, Z)}

*First negative:* obj(small, red, brick)

Update G: The current hypothesis in G (obj(X, Y, Z)) covers obj(small, red, brick), so specialize to hypotheses that do not cover obj(small, red, brick):

obj(large, Y, Z): Any large object. obj(X, white, Z): Any white object.

obj(X, blue, Z): Any blue object. obj(X, Y, ball)

obj(X, Y, ball): Any ball. obj(X, Y, cube): Any cube.

G: { obj(large, Y, Z), obj(X, white, Z), obj(X, blue, Z), obj(X, Y, ball), obj(X, Y, cube) }

*First positive:* obj(large, white, ball)

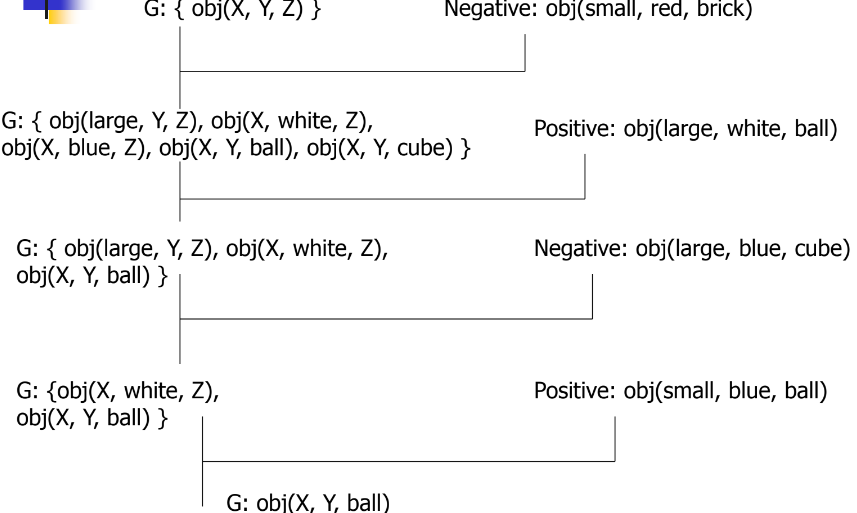
Update G: Remove hypotheses that do not cover obj(large, white, ball):

obj(large, Y, Z) covers obj(large, white, ball). obj(X, white, Z) covers obj(large, white, ball).

obj(X, Y, ball) covers obj(large, white, ball).

Remove obj(X, blue, Z) and obj(X, Y, cube) as they do not cover obj(large, white, ball).

G: {obj(large, Y, Z),obj(X, white, Z),obj(X, Y, ball)}

Next negative: obj(large, blue, cube)

Update G: Current hypotheses in G need to be specialized to exclude obj(large, blue, cube):

obj(large, Y, Z) covers obj(large, blue, cube) so specialize it:

obj(large, Y, ball): Any large ball.

obj(large, white, Z): Any large, white object.

Remove more specific hypotheses than others: obj(large, Y, ball) is removed.

G: {obj(X, white, Z),obj(X, Y, ball)}

*Next positive:* obj(small, blue, ball)

Final G: Remove hypotheses that do not cover obj(small, blue, ball):

obj(X, white, Z) does not cover obj(small, blue, ball), so it is removed.

G: {obj(X, Y, ball)}

**Algorithm for Searching in Version Space**

1. *Initialize:*

G with the most general description: { obj(X, Y, Z) }

S with the first positive example (ex+): { obj(small, red, ball) }

1. *For every learning example repeat:*
2. If positive (ex+, p):

Remove from G all hypotheses that don’t cover p. For each s∈S, if s doesn’t cover p:

Replace s with the most specific generalization that covers p.

Remove from S all hypotheses more general than others in S.

Remove from S all hypotheses more general than others in G.

1. If negative (ex-, n):

Remove from S all hypotheses that cover n. For each g∈G, if g covers n:

Replace g with the most general specialization that does not cover n.

Remove from G all hypotheses more specific than others in G.

Remove from G all hypotheses more specific than others in S.

1. Termination conditions:

If G = S and card(S) = 1, then a concept is found.

If G = S = { }, then there is no concept consistent with all hypotheses.

***Example***

Initial State: G: { obj(X, Y, Z) } S: { }

*First Positive Example:* obj(small, red, ball)

G: { obj(X, Y, Z) }

S: { obj(small, red, ball) }

*First Negative Example:* obj(small, blue, ball)

Remove from G: Nothing changes as obj(X, Y, Z) still covers all.

Remove from S: Nothing, as obj(small, red, ball) doesn’t cover obj(small, blue, ball).

*Second Positive Example:* obj(large, red, ball)

G: Remove hypotheses not covering obj(large, red, ball), resulting in { obj(X, red, Z) }

S: Generalize obj(small, red, ball) to cover obj(large, red, ball), resulting in { obj(X, red, ball) }

Second Negative Example: obj(large, red, cube)

G: Specialize {obj(X, red, Z)} to not cover obj(large, red, cube), resulting in {obj(X, red, ball)}

S: Remove any hypotheses covering obj(large, red, cube), but none do, so no change.

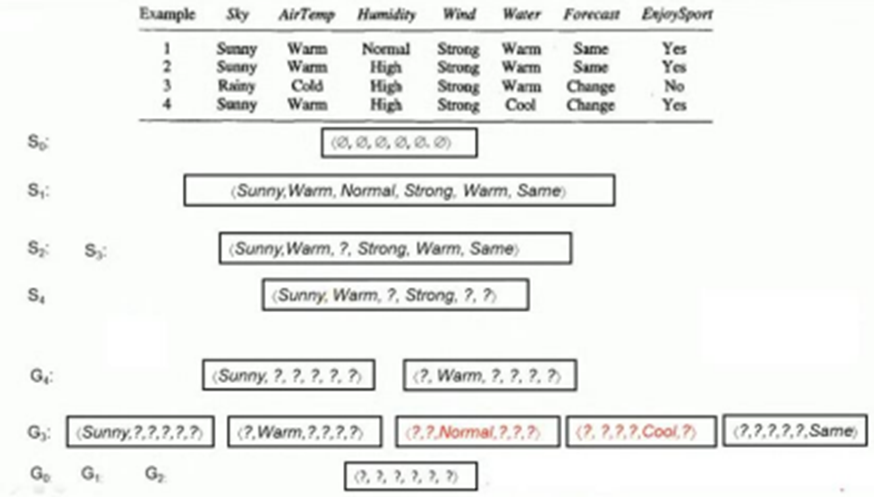
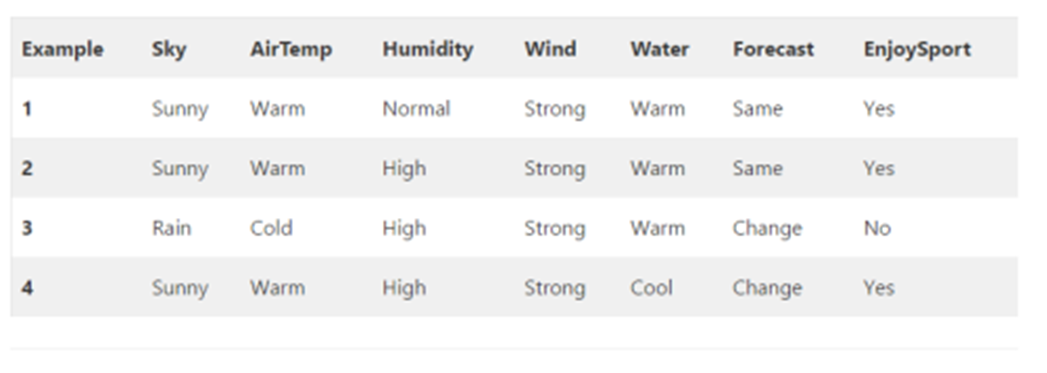
*Final State*

G: { obj(X, red, ball) }

S: { obj(X, red, ball) }

Outcome: Algo will continue to refine G and S until they converge.

***Example***



*First positive (Yes):* first check if hypothesis at the generic boundary is consistent with the input/training sample or not, then at the specific boundary.  
Compare G0 with first sample, All ‘?’ match sample1, +ve (yes) classification which is consistent with the label of sample1. G1 same as G0.

Compare S0 with the first sample. S0 has all null. No match = -ve classification which is not consistent with the label of sample1 (+ve/yes). Replace null with sample1.

*Second positive (Yes):* check at generic as well as specific.  
Compare G1 with 2nd sample, All question marks match sample2, hence the classification is positive(yes) which is consistent with the label of sample2. G2 same as G1   
Compare S1 with 2nd sample. Normal doesn’t match with High, -ve classification. Expected is positive. Replace Normal with ‘?’.

*Third negative (No):* check at specific (-ve & -ve, no change +ve & -ve, change) then generic.

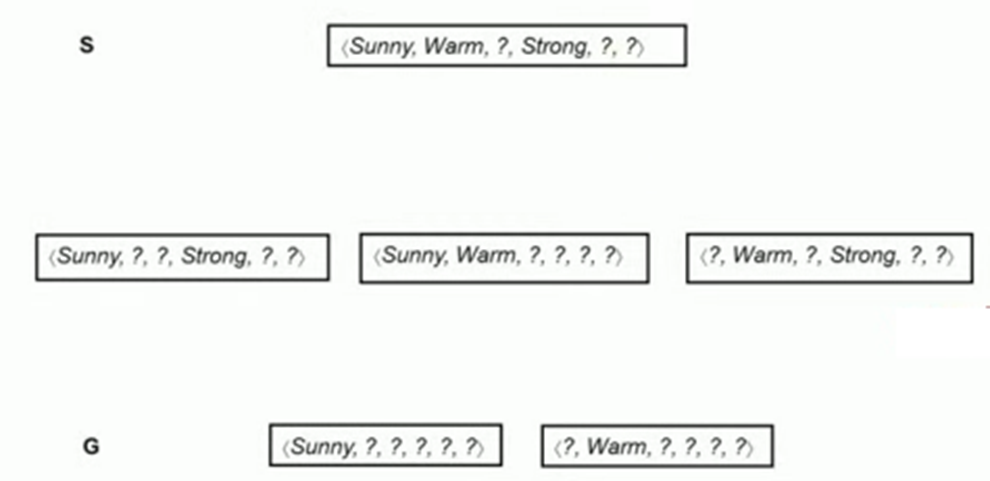
Compare S2 with 3rd sample, Sunny doesn’t match rainy, hence negative classification which is consistent with the ‘No’(-ve) of sample3. Retain. S3=S2.

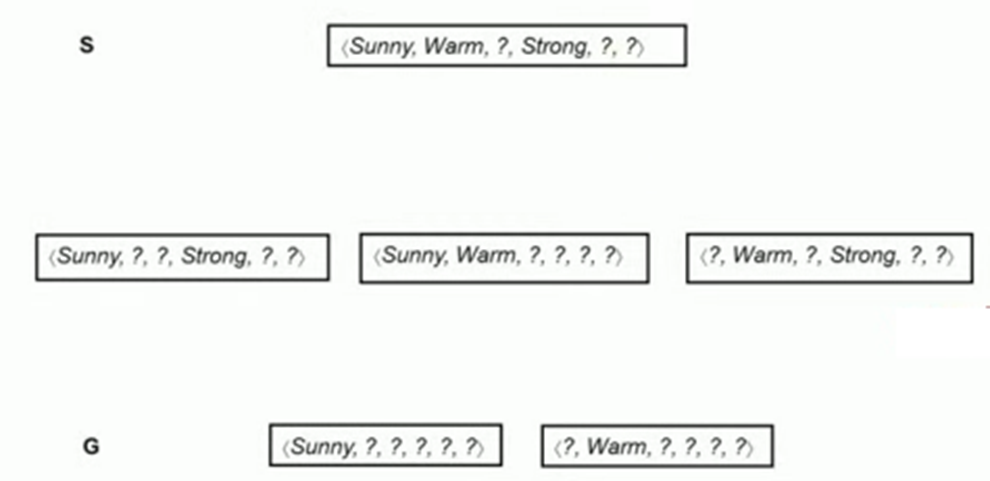
G2 is all ?s. Matches with sample3. Positive classification. Expected is negative (label of sample3). Not consistent. Substitute the opposite of Rainy in place of 1st ‘?’ Which is Sunny and rest all ‘?’ Will be retained. Now consider 2nd ‘?’ And repeat same. Retain consistent. Remove inconsistent (Normal, Cool).

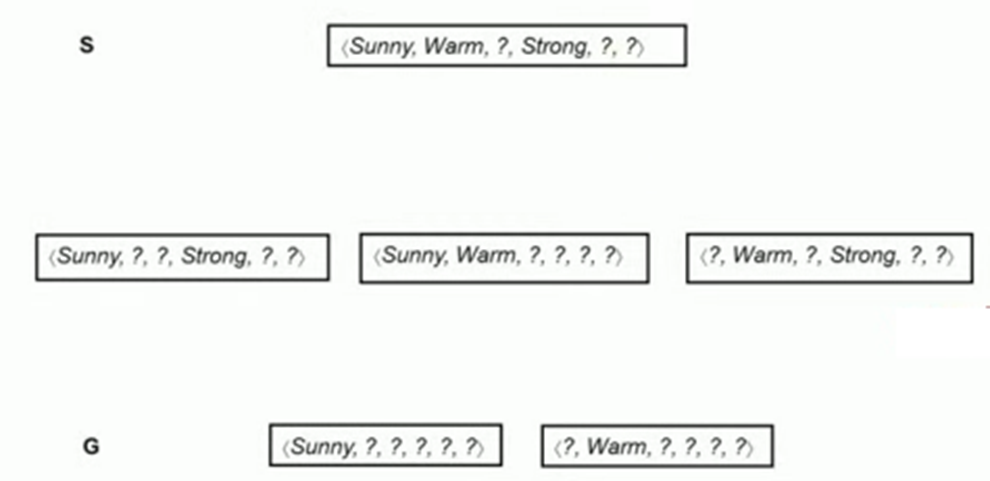
*Fourth positive (Yes):* check at generic as well as specific.

Compare all hypotheses in G3 with the training sample4. Retain those in G4 that are consistent with Sample4 and remove other inconsistent hypotheses.  
Compare S3 with sample4. Sunny, warm matches. ‘?’ matches with high. Warm doesn’t match Cool and Same doesn’t match change. Replace Warm & Same in S3 with ? to get S4.

*Learned Version Space by Candidate Elimination Algorithm*







All training samples are over

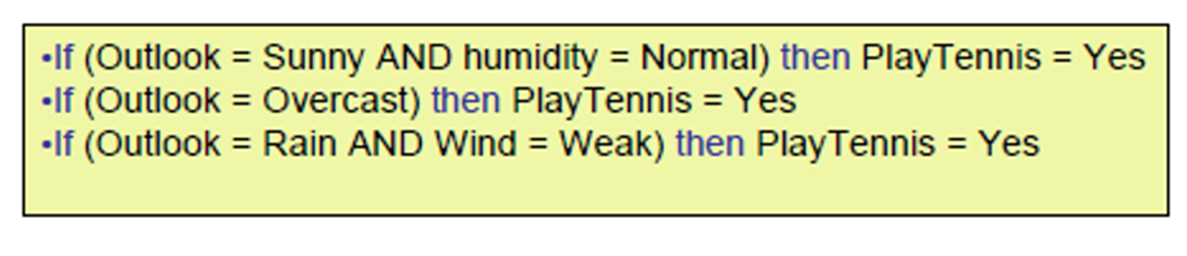
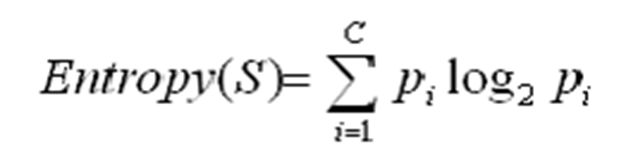
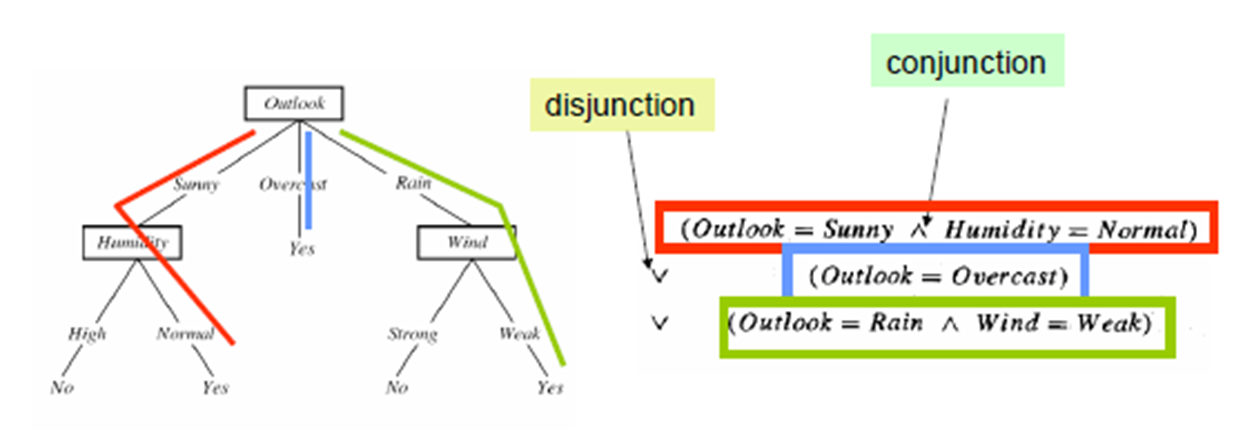
S4 and G4 not same. There is more than one hypothesis. So we need to write a few more hypotheses considering S4 and G4.

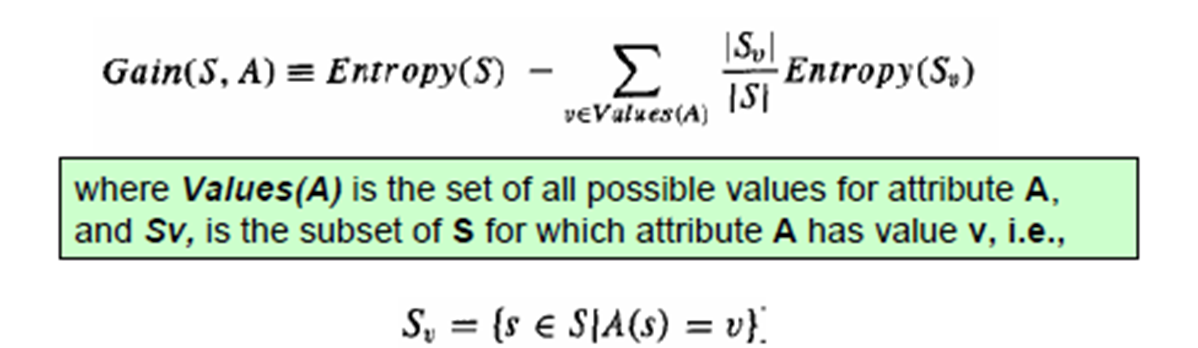
Warm and ‘?’ do not match so replace ‘?’ by Warm etc. Here consistent hypotheses are 6.

**Decision Trees** are used to classify data by splitting it based on attribute values.

**ID3** is a method for building decision trees

Top-Down Approach: Builds the tree from the root to the leaves without backtracking.





**Entropy** measures the impurity or disorder within a dataset.

A completely homogeneous sample (all instances belong to one class) has an entropy of 0.

An equally divided sample (instances are evenly split among classes) has an entropy of 1.

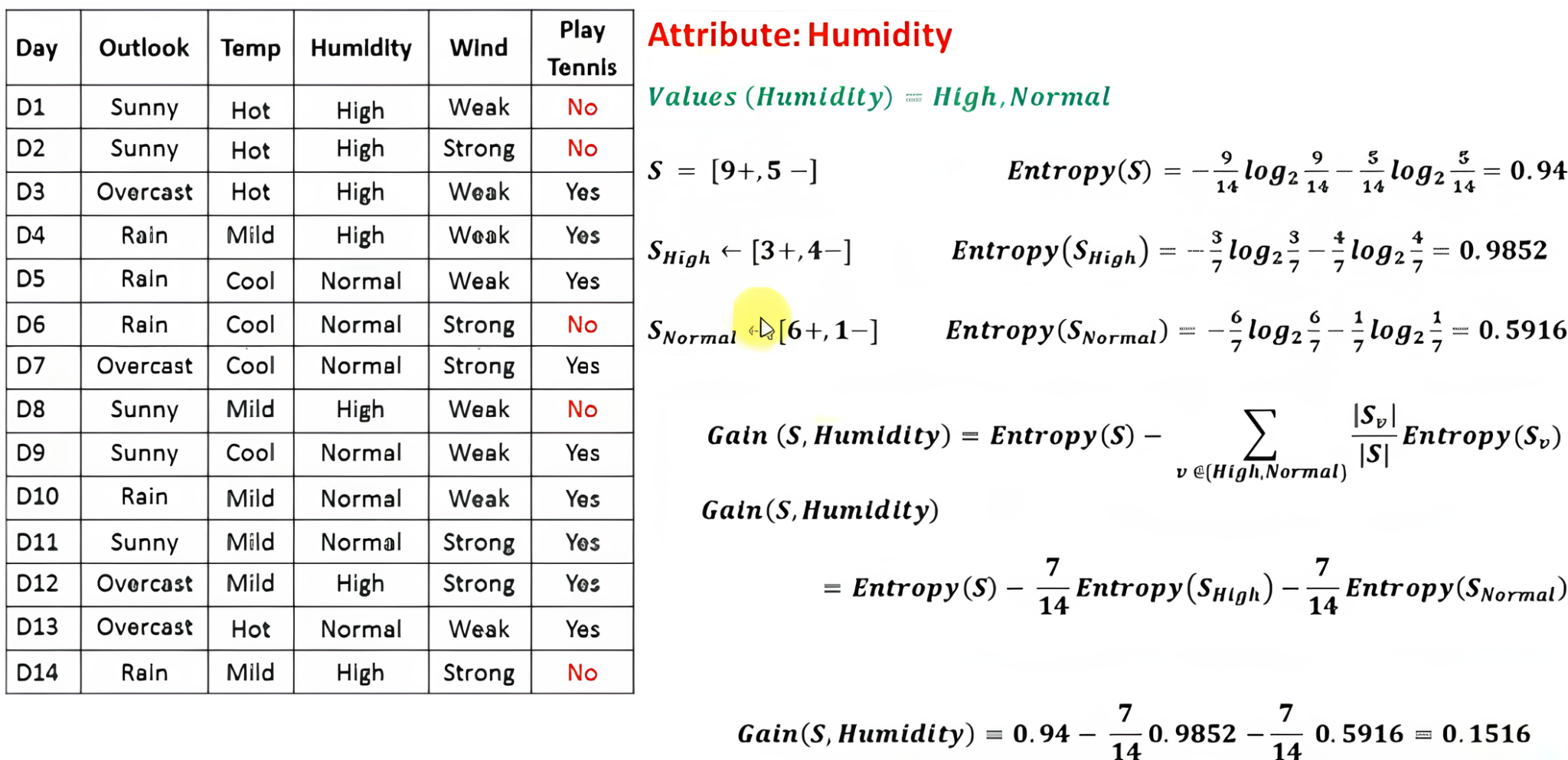
**Information Gain (IG)** is used to decide which attribute to split the data on at each step in building the decision tree. It measures the reduction in entropy from the original dataset to the subsets formed by splitting on an attribute.

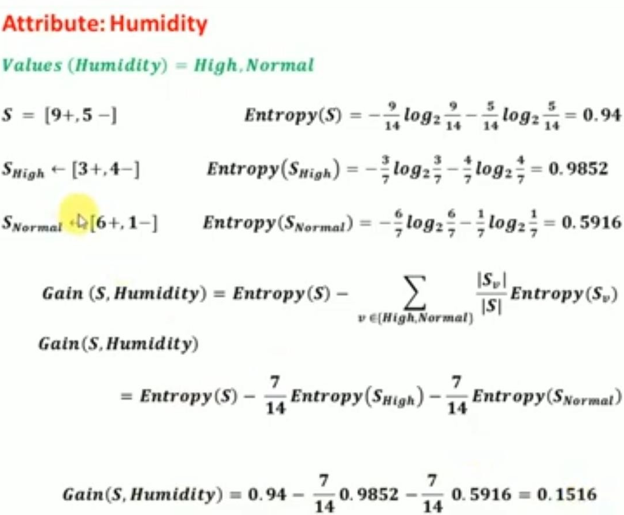
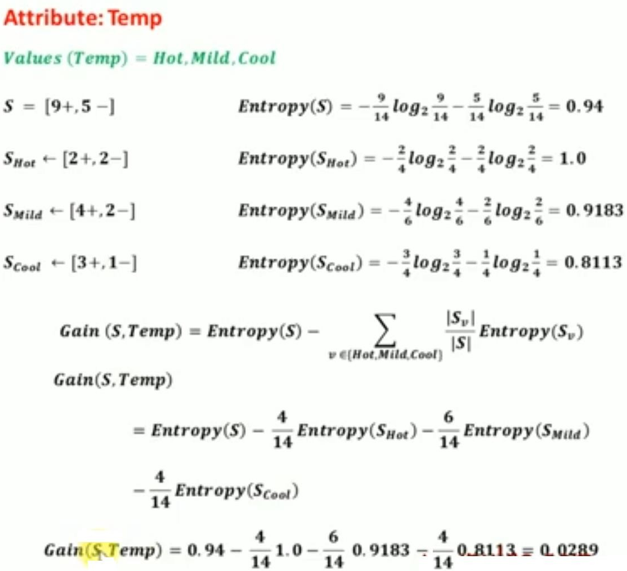
**Advantages of Using ID3**

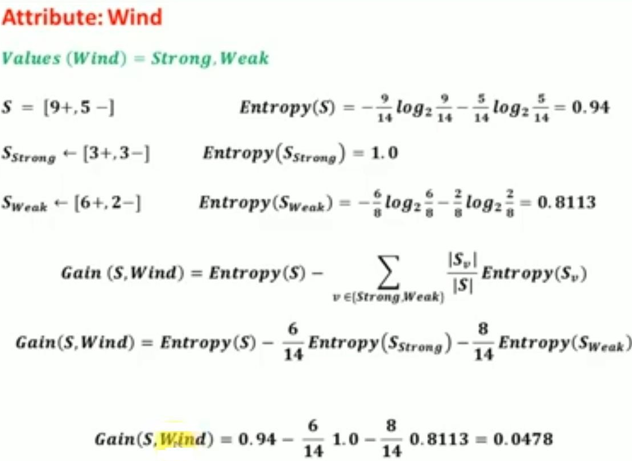
1. *Understandable Rules:* Creates clear if-then-else rules.
2. *Fast Construction:* Quickly builds the tree.
3. *Short Trees:* Produces concise trees.
4. *Efficient Testing:* Tests only necessary attributes.
5. *Pruning:* Reduces tests by identifying leaf nodes.
6. *Exhaustive Search:* Uses the entire dataset for optimal splits.

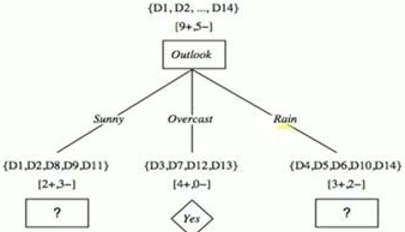
**Disadvantages of Using ID3**

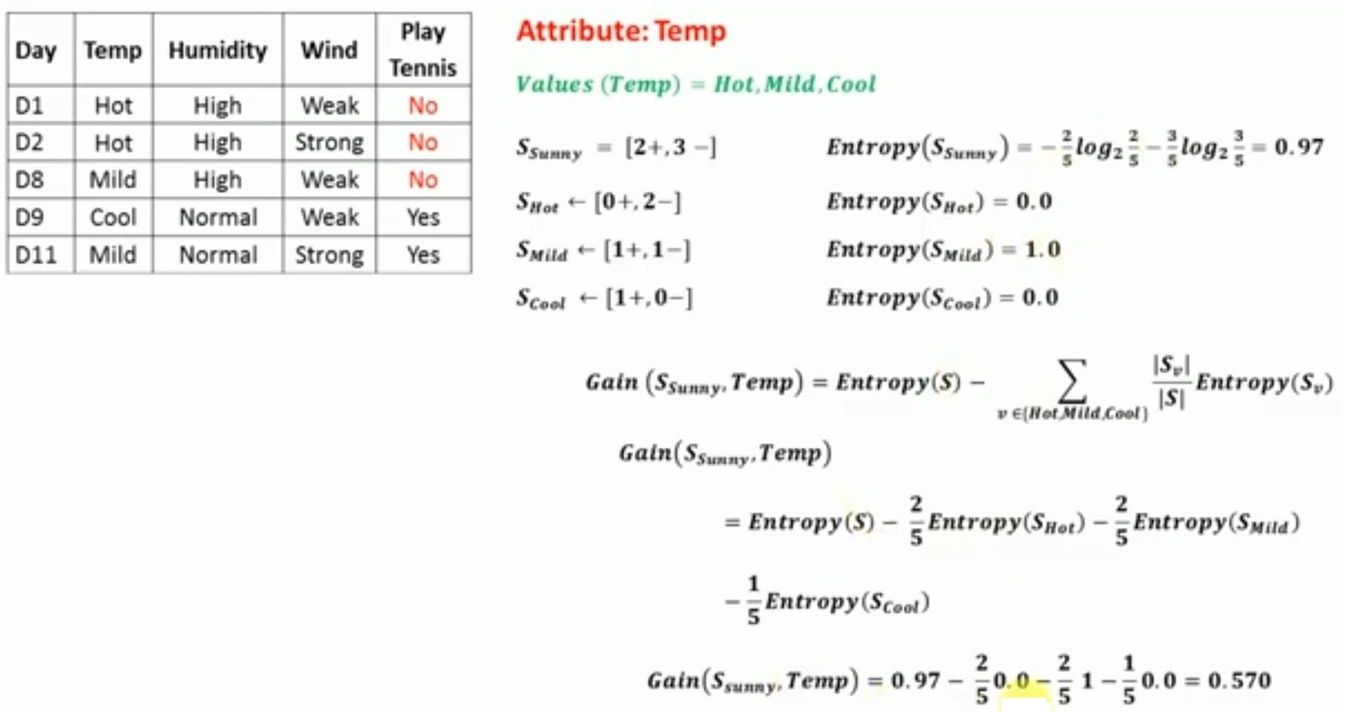
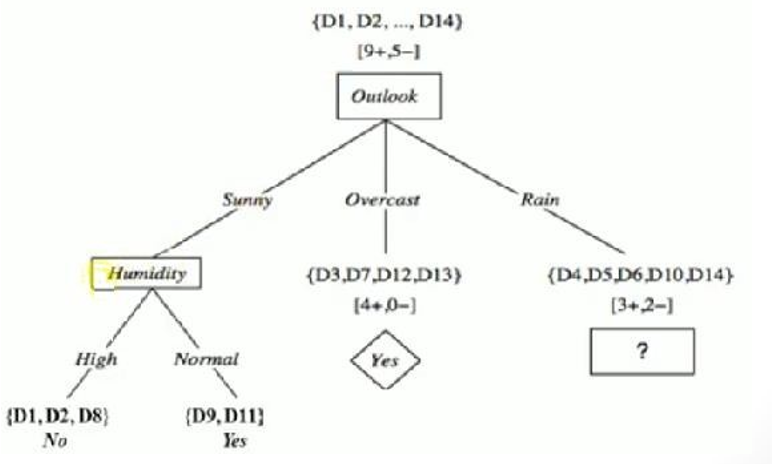
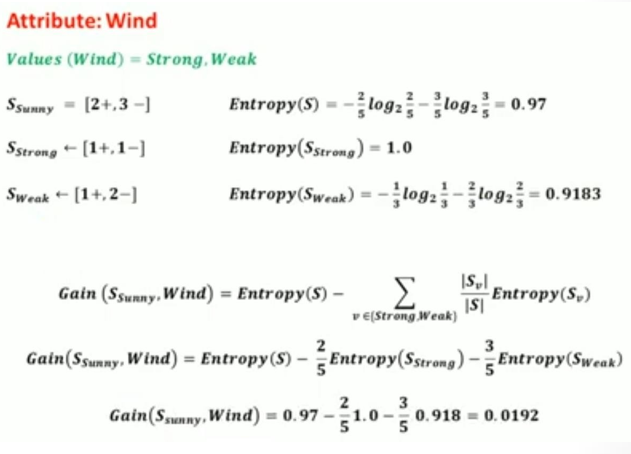
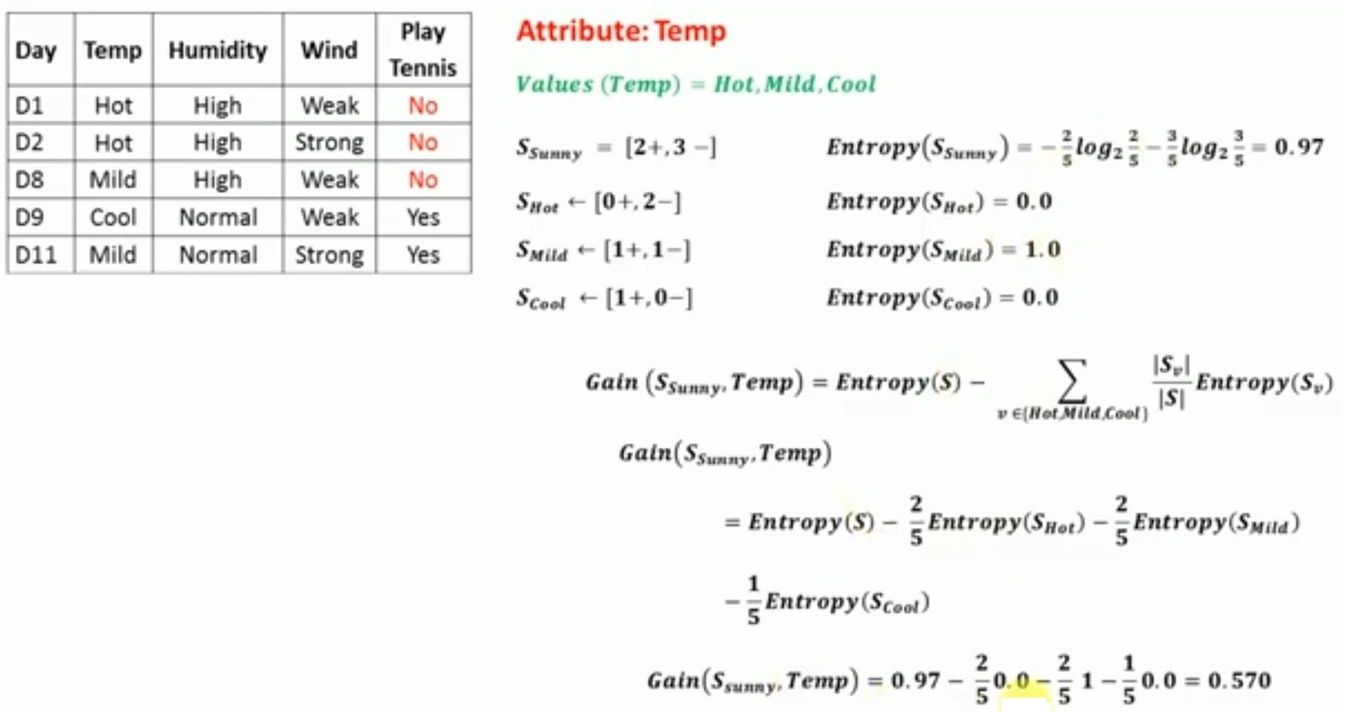
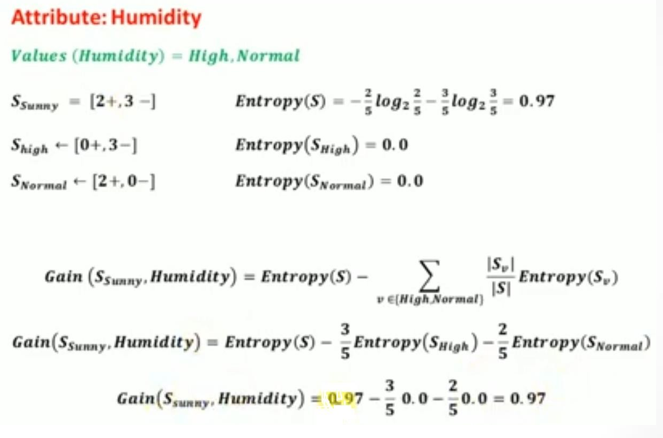
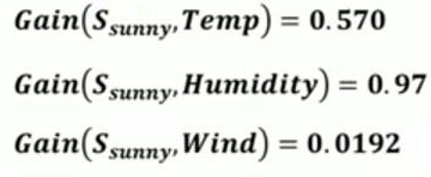
1. *Overfitting:* Can overfit with small datasets.
2. *Single Attribute Focus:* Tests one attribute at a time.
3. *Continuous Data:* Computationally expensive for continuous data.





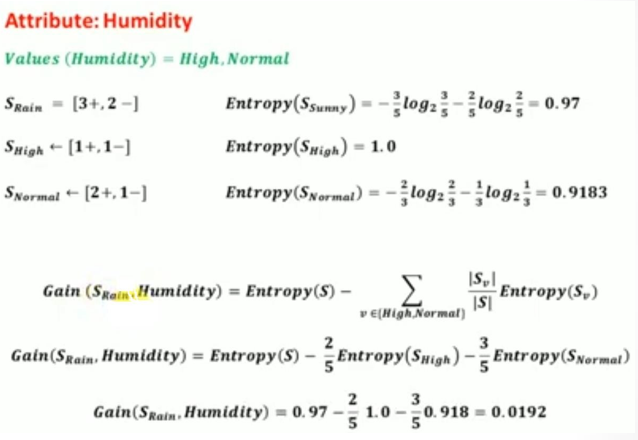
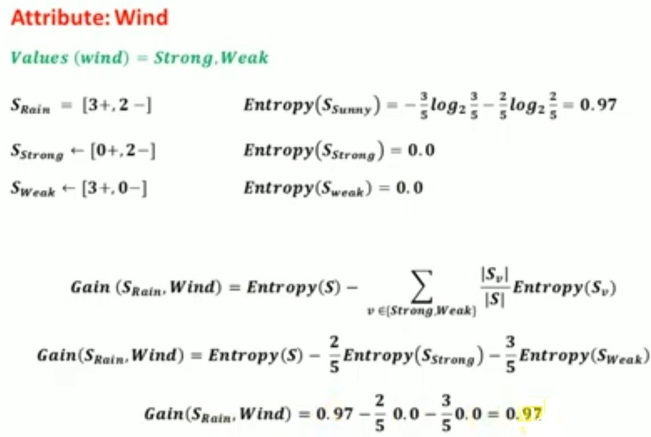
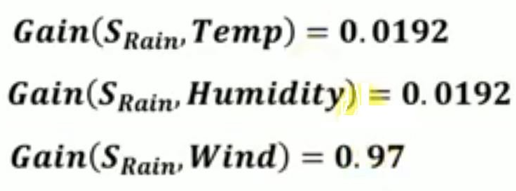


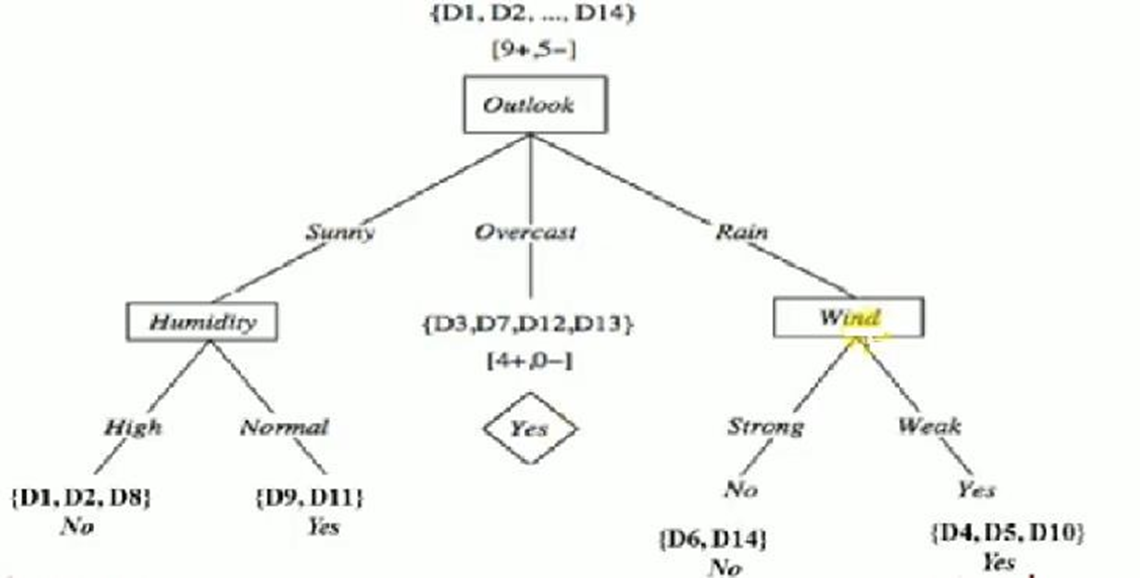


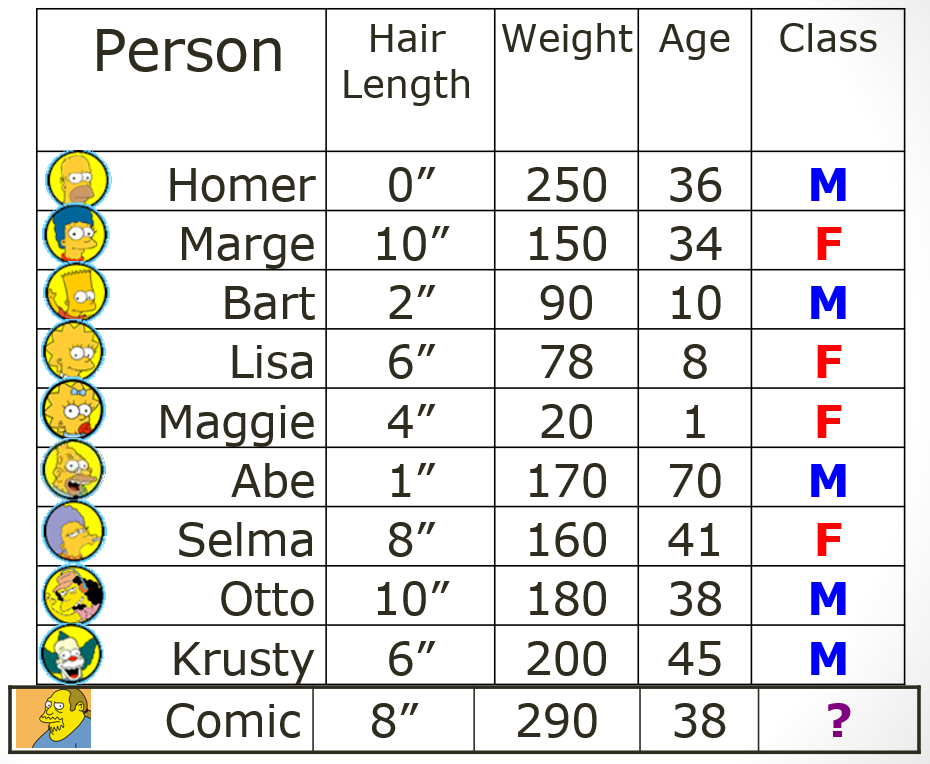
 

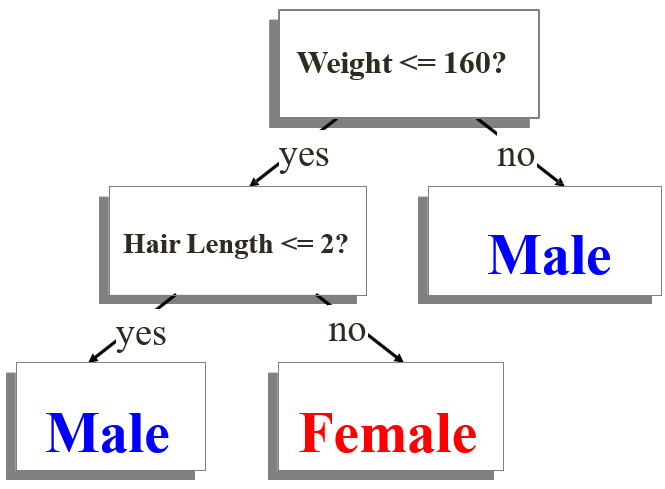


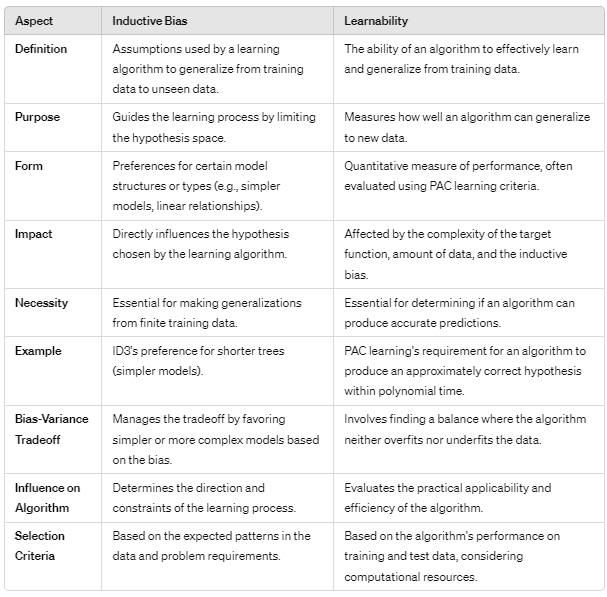












**Information**, facts, and skills acquired through experience or education, which ML algorithms use to make predictions or decisions.

1. *Explicit Knowledge:* Clearly defined and easily communicated, such as rules, procedures, and instructions.
2. *Implicit Knowledge:* Gained through experience and often harder to communicate, such as intuition and know-how.

**Supervised Learning**

A type of machine learning where the model is trained using labeled data. Each input data point is paired with a correct output label. Training a model to recognize cats and dogs using a dataset of images labeled as "cat" or "dog." Process:

1. *Training Data:* Consists of input-output pairs.
2. *Model Training:* The algorithm learns to map inputs to outputs using the training data.
3. *Prediction:* The trained model predicts outputs for new, unseen inputs.

***Common Algorithms:***

1. Linear Regression
2. Decision Trees
3. Support Vector Machines (SVM)
4. Neural Networks

***Examples:***

1. Email spam detection
2. Image classification
3. Sentiment analysis

***Advantages:*** Clear and measurable performance. Easier to understand and interpret the results.

***Disadvantages:*** Requires a large amount of labeled data. Can be expensive & time-consuming to label data

**Unsupervised Learning**

A type of ML where the model is trained using unlabeled data. Algo tries to find patterns or groupings within data without any predefined labels. Grouping customers into diff segments based on purchasing behavior w/o knowing specific categories in advance. Process:

1. *Training Data:* Consists of input data without output labels.
2. *Pattern Recognition:* Algo identifies patterns, correlations, & structures within data.
3. *Grouping:* The algorithm may group data into clusters or reduce data dimensionality.

***Common Algorithms:***

1. K-means Clustering
2. Hierarchical Clustering
3. Principal Component Analysis (PCA)
4. Autoencoders

***Examples:***

1. Customer segmentation
2. Anomaly detection
3. Market basket analysis

***Advantages:*** Can work with unlabeled data, which is more readily available. Useful for discovering hidden patterns and intrinsic structures in data.

***Disadvantages:*** Results can be harder to interpret and evaluate. No explicit feedback on the accuracy of the model's predictions.

**Reinforcement Learning** is a feedback-based machine learning technique where an agent learns to behave in an environment by performing actions and receiving feedback. Positive feedback reinforces good actions, while negative feedback penalizes bad actions.

Key Characteristics:

1. *No Labeled Data:* The agent learns from experience without labeled data.
2. *Sequential Decision Making:* RL is suited for problems where decisions are made in sequence, aiming for long-term goals.
3. *Exploration and Exploitation:* The agent interacts with the environment, exploring and learning from feedback to maximize rewards.

Ex: A robotic dog learning to move its arms or AI agent navigating a maze to find a diamond.

***Core Concepts***

1. *Agent:* The entity that interacts with the environment.
2. *Environment:* Situation in which agent operates, assumed to be stochastic (random).
3. *Action:* Moves or decisions taken by the agent.
4. *State:* The situation or condition returned by the environment after an action.
5. *Reward:* Feedback received from the environment to evaluate the agent's actions.
6. *Policy:* Strategy the agent uses to determine its next action based on current state.
7. *Value:* Expected long-term return with a discount factor, considering future rewards.
8. *Q-value:* Similar to value but includes the current action as an additional parameter.

***Key Features***

1. *Exploration:* The agent is not pre-programmed & learns by exploring the environment.
2. *Hit and Trial:* Learning through a process of trial and error.
3. *State Transitions:* The agent changes states based on actions and feedback.
4. *Delayed Reward:* Rewards may not be immediate.
5. *Stochastic:* Environment = random, requiring agent to explore for optimal rewards.

***Approaches to Implement Reinforcement Learning***

1. *Value-based:* Find optimal value function, maximizing long-term return at any state under any policy. Q-learning, where agent learns value of actions in different states.
2. *Policy-based:* Find the optimal policy for maximum future rewards without explicitly using the value function. Deterministic Policy: Always produces the same action for a given state. Stochastic Policy: Uses probabilities to determine actions.

Example: REINFORCE algorithm, which directly learns the policy.

1. *Model-based:* Create a virtual model of the environment and use it to learn. Varies for each environment as the model representation differs. Example: Dyna-Q, which combines model-based planning with Q-learning.

**Prediction Error:** The diff b/w actual values & predicted values generated by a model.

1. *Training Error:* Error on the training dataset.
2. *Test Error:* Error on a new, unseen dataset.

**Bias Error:** The error due to overly simplistic assumptions in the learning algorithm. It represents how much the average model prediction differs from the actual values. High bias can cause underfitting, where model is too simple to capture the underlying pattern of data.

Example: A linear model trying to fit a quadratic relationship.

**Variance Error:** The error due to the model's sensitivity to small fluctuations in the training data. It represents how much the model's predictions vary for different training sets. High variance can cause overfitting, where model captures noise along with underlying pattern.

Example: A high-degree polynomial model fitting random noise in the data.

**Irreducible Error:** The error inherent in the data itself due to noise, randomness, or unmeasured variables. This error cannot be reduced by any model. Represents the limit of a model's prediction accuracy.

**The Bias-Variance Trade-off:** The balance between bias and variance errors in a model. Adjusting the model to decrease one often increases the other. Find the right complexity level for the model that minimizes the total prediction error, a combination of bias, variance, and irreducible errors.

Total Error = Bias^2 + Variance + Irreducible Error

1. *Underfitting:* High bias and low variance.
2. *Overfitting:* Low bias and high variance.

**Intro to Fitting:**

1. *Underfitting:* When a model is too simple and fails to capture the complexity of the data, leading to high bias and poor performance on both training and test data. Example: Using a straight line to fit non-linear data.
2. *Overfitting:* When a model is too complex and captures the underlying pattern and noise in the training data, leading to high variance and poor generalization to new data.Example: Using a high-degree polynomial to fit data that is inherently linear.
3. *Proper Fitting:* When a model appropriately captures the underlying pattern without fitting the noise, leading to a good balance between bias and variance, and performing well on both training and test data. Example: Using a polynomial of appropriate degree to fit a quadratic relationship in the data.