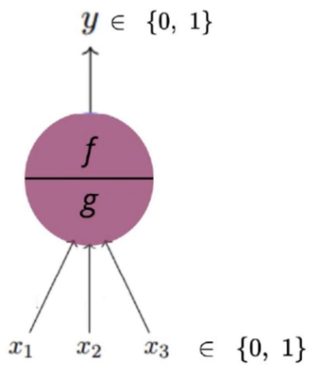
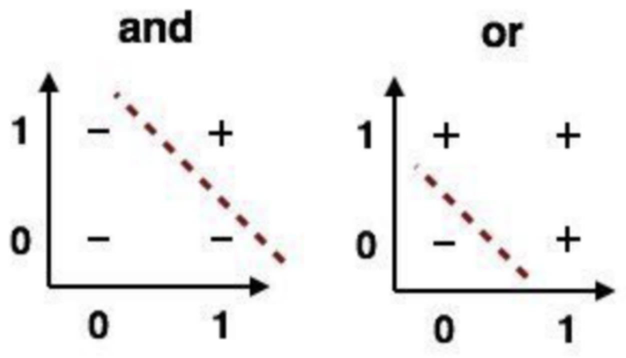
Topics required to prepare for the mid-term (March 18)
Closed-book and on paper

1. Warren McCulloch and Walter Pitts neuron

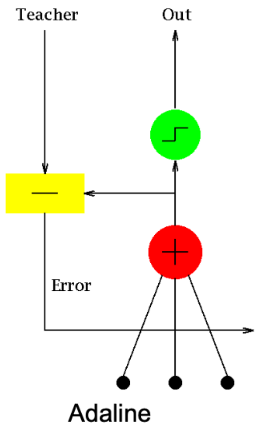
- First generally recognized AI work
- Artificial neurons are characterized as being “on” or “off”
- Any computable function could be computed using some network of connect neurons, all logical connectives (AND, OR, NOT…) could be implemented by simple network structures





2. Perceptron

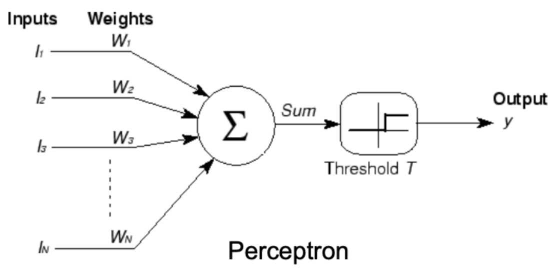
- The learning algorithm can adjust the connection strengths of a perceptron to match any input data, provided such match exists
- A perceptron could not be trained to recognize when its inputs were different, such as XOR gate



Difference between adaline and perceptron is in the learning phase:

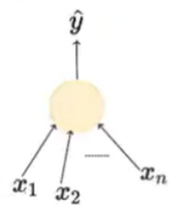
In adaline, weights are adjusted according to the weighted sum of the inputs.

In perceptron, the net is passed to the activation function and the function's output is used for adjusting the weights.



3. Differences between Warren McCulloch and Walter Pitts neuron, and perceptron

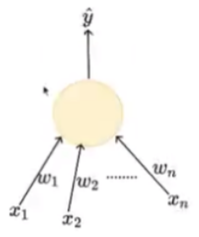
MP Neuron



$$\hat{y} = 1 \text{ if } \sum_{i=1}^{n} x_i \geq b$$
$$\hat{y} = 0 \text{ otherwise}$$

- ☹ Boolean inputs
- ☹ Linear
- ☹ Inputs are not weighted
- ☺ Adjustable threshold

Perceptron



$$\hat{y} = 1 \text{ if } \sum_{i=1}^{n} w_i x_i \geq b$$
$$\hat{y} = 0 \text{ otherwise}$$

- ☺ Real inputs
- ☹ Linear
- ☺ Weights for each input
- ☺ Adjustable threshold

4. Main reasons for the success of artificial intelligence today

- Advances in computing power (hardware): faster processors
- World Wide Web and very large datasets: data explosion
- Reinvention of the back-propagation learning algorithm and other techniques: algorithmic advancements

5. Risks and benefits of AI

Importance of governance and regulation

+ Free humanity from repetitive work
+ Accelerating scientific research
+ Increase production of goods
+ Cybersecurity: Defend against cyber attacks by recognizing unusual patterns of behavior
- Lethal autonomous weapons
- Surveillance and persuasion (mass surveillance of individuals and detect activities of interest - illegal)
- Biased decision making: biased by race, gender,...
- Impact to employment: machines replacing humans, disruption to employment
- Safety-critical applications: difficult to verify AI system's choices, need to develop technical and ethical standards

6. Search algorithms

- Takes a problem as input and returns a solution, or an indication of failure.
- Node = state; edge = action; root of tree = initial state
- Action cost function: gives a numerical cost of applying actions to search and reach goal state.
- The total cost of the path is the sum of individual action costs.
- When state space is very large or infinite, we need a local search.
- Problems: fully observable, deterministic, static environments. Save the path to reach the solution.
- Optimal solution: when it finds the solution with the lowest cost.

a. Uninformed

Agent has no information about how far it is from the goal.

i. Depth-first search

- LIFO queue
- Proceeds to the deepest level on the search tree
- Not cost-optimal, returns the first solution it finds, even if it's not the cheapest solution
- Complete in finite state spaces. Can get stuck in infinite state spaces, thus incomplete.
- Space complexity: O(bm) , b = branching, m = max depth

ii. Breadth-first search

- FIFO queue
- The root is expanded first, then all its successors next, then their successors, and so on.
- Post optimal: chooses the best solution among all options
- Complete
- Time and space complexity: $O(b^d)$
- Good for small instances

iii. Uniform cost search

- Spreads out in waves of uniform path-cost
- It keeps track of the cost from the initial state, compares costs, keeps bigger on hold, moves on the next from the least cost direction. On each step it expands and compares again.
- Cost optimal: consider all options with low cost before exploring paths w/ high cost but perhaps useful options
- Complete
- Time and space complexity: $O(b^{\text{max steps to reach solution}})$ can be $> O(b^d)$

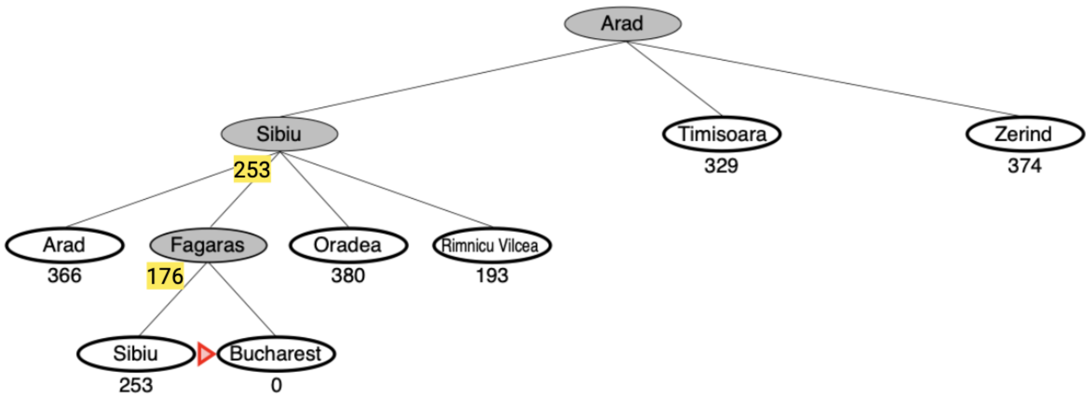
b. Informed

Agent uses some information to make a decision on where to go next, it can estimate how far it is from the goal.

Heuristic Function h(n) = estimated cost to reach the goal from n.

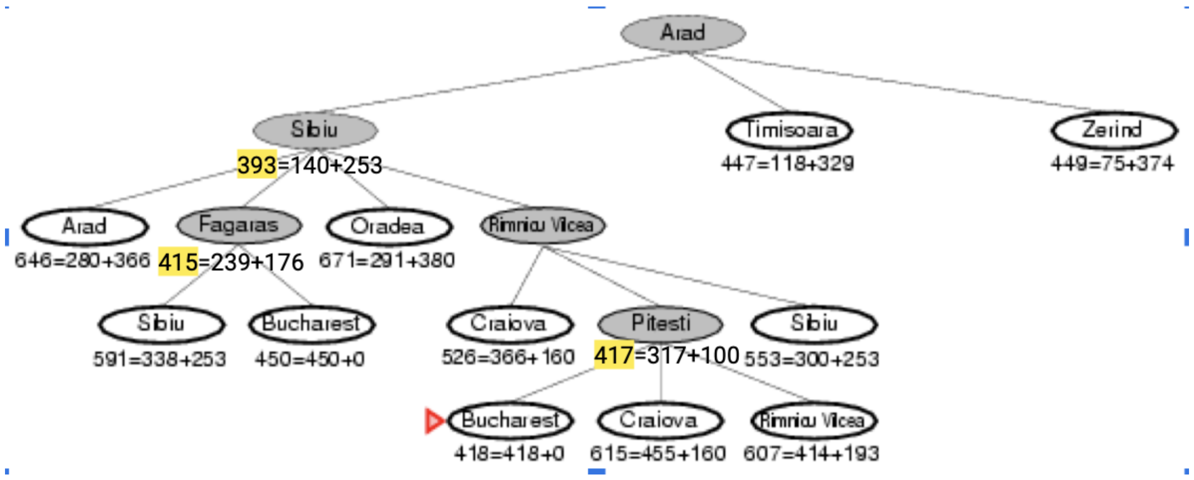
i. Greedy best-first search

- Complete in finite state spaces.
- Is a form of best-first search. It chooses to expand to the next node with the lowest h(n) value, which seems to be the closest point to reach the goal.
- Does not always find the optimal solution (with the optimal cost). But it is often efficient.
- On each iteration it tries to get closer to the goal. It does not backtrack.



ii. A^* search

- Complete and optimal.
- It may be cost-optimal depending on the heuristic function.
- The most common informed search algorithm.
- It's a form of best-first search with a combination of uniform search and greedy algorithm, where the cost from the current state to the origin g(n) and to the goal h(n) are known.
- From the initial state it goes on through the lowest cost path, but it keeps track of the last iteration costs. It expands from the lowest cost, and then it compares the costs again, always choosing to expand from the lowest cost node.



7. Local search algorithms (discrete space)

- Start searching from the initial state and expand to its neighbors.
- It does not record the path or the set of states it visited.
- It finds a solution, not always optimal.
- These algorithms have the advantage of using little memory (not storing the path), and are able to find good solutions even in large or infinite state spaces.

a. Hill-climbing

- Used to find the global maxima. Compare values of left and right neighbors, pick the best value using the objective function, and then compare with the current node using the objective function. Do not look beyond the neighbors.
- Goal is to find the highest peak, the global maximum
- Keeps track on the current state and moves to the neighbor with the highest value.
- It may get stuck in a local maxima, when it reaches a ridge, or a plateau
- Reasonable output, able to find a good local maxima after a small number of restarts.

The steps to find the solution:
Choose a random initial state and assign it to current
Loop
Select the better neighbor of the current state
If looking for a maximum global: $f(N) \le f(C)$: return C (it's the best)
$f(N) > f(C)$: update C with N

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← problem.INITIAL
    while true do
        neighbor ← a highest-valued successor state of current
        if VALUE(neighbor) $\leq$ VALUE(current) then return current
        current ← neighbor
```

a.3 Random-restart Hill Climbing

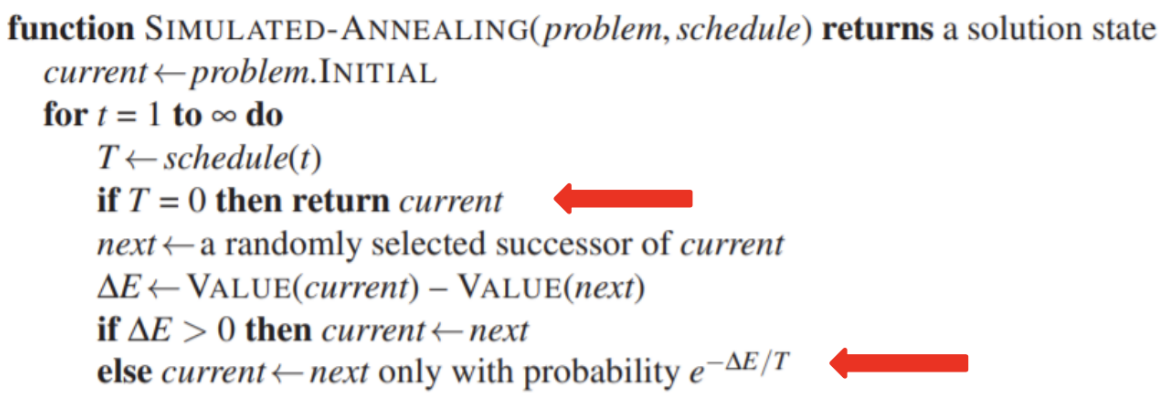
- Tries random successors until it finds a better one for x iterations.
- Search next looking at random states, not neighbors.

b. Simulated annealing

- Complete and efficient.
- Similar to Stochastic Hill-Climbing, but it accepts some downhill moves.
- Combines hill-climbing with random walk.
- Starts at a random state.
- Schedule a temperature T
- Compute ΔE = value (current) - value (next)
- If the next random choice has a better value, it always updates the current state, or it uses probability to update the state.
- $P = e^{-\Delta E / T}$
- If probability is high P ↑ , it moves around more (more randomness of current updates)
- T ↑ = P ↑
- ΔE ↓ = P ↑
- Make some “shakes” (using T) to allow the state to change if it gets stuck. At the beginning, the shakes are more intense, allowing for bigger moves, and gradually reduces the intensity of the shakes.
- Gets better results because it keeps adjusting T
- Goal is to find highest or lowest point (global max or global min)

The steps to find the solution:

1. Assign a high T value
2. Choose a random initial state and assign it to current
3. Loop
4. Select a random next state
5. compute $\Delta E = |\text{value (C)} - \text{value (N)}|$
6. If looking for a minimum global: if $\Delta E > 0$: C = N
   else : if T is high, high P of updating C with N
   if ΔE is low, high P of updating C with N



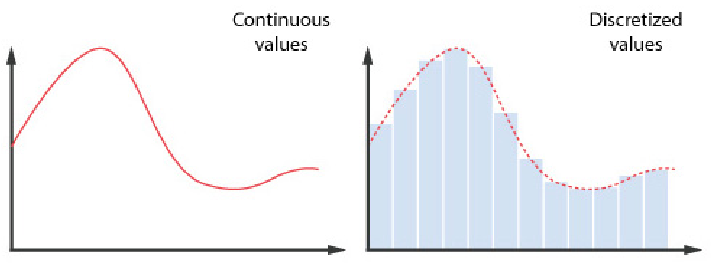
```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  current ← problem.INITIAL
  for t = 1 to ∞ do
    T ← schedule(t)
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE(current) – VALUE(next)
    if ΔE > 0 then current ← next
    else current ← next only with probability $e^{-\Delta E/T}$
```

8. Local search algorithms (continuous space)

Continuous spaces have infinite branching factors, so they cannot be solved by the previous algorithms.

a. Discretization

- Limit the space into a grid with spacing of size δ (lowercase delta).
- Once it is discretized, apply a discrete algorithm.



b. Equating gradient to zero

- $\nabla f = 0$ (Gradient)
- Use calculus to solve a problem analytically
- Find a x where ∇f (x) = 0 to find a maximum or minimum

  b.1. Compute the gradient locally
- x ← x + α ∇f (x)
- α is the step size. If it's too big, it may miss the global maxima; if it's too small, it may need too many steps.
- ∇f (x) tells if the slope is positive or negative
- Compute the derivative f '(x)=0 to find the max /min
- ∇f (x) computes multiple values of x
- f '(x) computes function of only one value of x

- Where the slope is zero, it's where you find the plateau
- Advantage: simple to update equation
- Disadvantage: difficult to choose the step size α
- Use gradient descendant if the sample size is big

i. Gradient ascent/descent

- Gradient ascendent : $x \leftarrow x + \alpha \nabla f(x)$, goal towards the max, same direction of gradient
- Gradient descendent : $x \leftarrow x - \alpha \nabla f(x)$, goal towards the min, opposite direction of gradient
- $\nabla f(x)$ a vector that gives direction/slope, α is the step size
- $\nabla f(x) < 0$, negative gradient
- $\nabla f(x) = 0$, min or max
- $\nabla f(x) > 0$, positive gradient

9. Quantifying uncertainty

Uncertainty and Rational Decisions: to make choices, an agent must have preferences of the possible outcomes. Use probability to help quantify uncertainty.

a. Unconditional (prior) and conditional (posterior) probability

Unconditional or Prior Probability: when you don't have any evidence, consider all possibilities.

$$P(a) = 1 - P(\neg a)$$

Conditional or Posterior Probability: when you have some information (evidence), a "given", to compute the probability of getting that outcome.

Given any events a and b:

$P(a \mid b) = \frac{P(a \cap b)}{P(b)}$, for P(b) > 0 $\rightarrow$ Probability of a reduced to the space of b.

$P(b \mid a) = \frac{P(a \cap b)}{P(a)}$, for P(a) > 0 $\rightarrow$ Probability of b reduced to the space of a.

$P(a \cap b) = P(a|b)\,P(b)$ $\rightarrow$ rearrange terms in the same equation

b. Inference using joint probability distribution

Probability distribution: assign weights to outcomes (the probability of getting that outcome). The probability may change given some evidence.

Full Join Distribution: a simple method for computing probabilities given observed evidence using the "knowledge base" - a table with all variables and their combined data. To compute the unconditional probability of a single variable a, use $P(a) = 1 - P(\neg a)$. To compute the conditional probability of a single variable a given b, use $P(a \mid b) = \frac{P(a \cap b)}{P(b)}$.

c. Independence

For conditional, unconditional probabilities, and joint distributions we can have **independence**, which means that given some evidence won't change the probability. Independence: assume all effects are independent of each other

$$P(\text{effect}_1 \mid \text{cause})\, P(\text{effect}_2 \mid \text{cause}) \ldots P(\text{effect}_N \mid \text{cause})$$

d. Bayes rule

Bayes' Rule: just using product rule, where

$P(a,b) = P(a|b)\,P(a)$ and $P(a,b) = P(b|a)\,P(b)$:

$$P(b \mid a) = \frac{P(a \mid b)\,P(b)}{P(a)},$$

where it's possible to compute a single term in terms of the three other terms. It's very used on the medical field for diagnosis such as

$$P(disease \mid symptom) = \frac{P(s|d)\,P(d)}{P(s)}$$

$$P(\text{cause} \mid \text{effect}) = \frac{P(\text{effect}|\text{cause})\ P\ (\text{cause})}{P(\text{effect})}$$

e. Naive Bayes

Naive Bayes = Bayes' Rule + independence

it's a combination of Bayes' Rule (product rule) and independence

$$P(\text{cause, effect1, effect2 ,... effectN}) = \frac{P(e1 \mid \text{cause})\ P(e2|c)\ P(eN|c)\ P(\text{cause})}{P(e1)\ P(e2)...P(eN)}$$

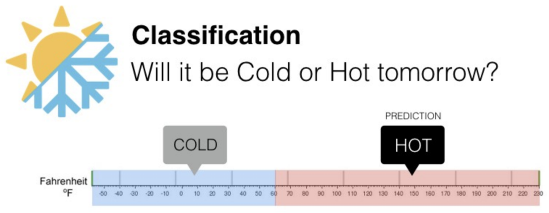
Naive Bayes uses conditional probability to compute unknown probabilities, assuming independence of all effect variables. It can be used for text classification.

Step1: Compute prior probabilities (of the whole training dataset for the expected outputs). Ex: P(positive) = # positive / total docs
Step2: Compute posterior probabilities (conditional) for each word. P(given word | positive) = # occurrences of given word in positive subset / # total words on positive subset
Step3: Analyze new words using the words with highest posterior probability, considering all words are independent.

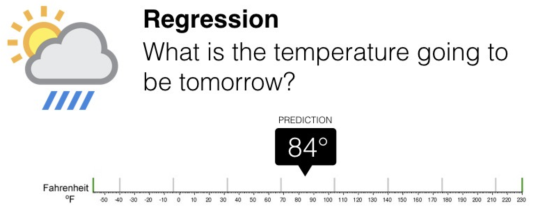
To avoid zero posterior probability, perform **laplace smooth**, adding 1 into the numerator and adding the total number of unique words of the training dataset into the denominator.
Ex. P(given word | positive) = (# occurrences of given word + 1) / (# total words on positive subset + total unique words of training dataset)

10. Learning from examples

a. Classification vs regression



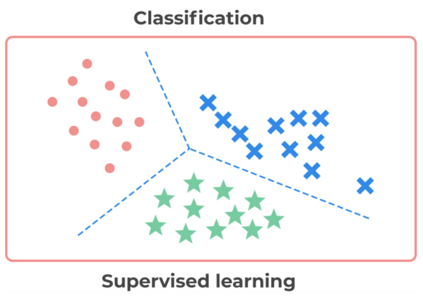
- Classification
  - Predicts different classes or categories
  - Predicts qualitative output of an input
  - All output values must form a finite set
  - Uses a function to divide data into a finite number of classes /categories

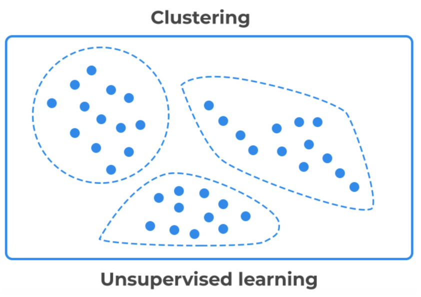


- Regression
  - Predicts continuous values
  - Predicts quantitative output of an input
  - All output values must form an infinite set
  - Uses a function to predict a continuous value

b. Supervised learning vs unsupervised learning

- Supervised
  - Agent learns from labeled examples
  - Train using labeled input/output pairs
  - Classifies similar inputs based on features
  - Categories are known by developer and by model





- Unsupervised
  - Agent finds patterns on the data to cluster them together
  - Train using inputs with no output labels
  - "Cluster" inputs together based on features that model finds relevant
  - Model outputs clusters of similar inputs, not knowing what they are
- Reinforcement Learning : learn from mistakes

c. Choosing a good model (Bias and Variance)

- Bias : how accurately we can fit the training data in the function
  - High Bias = UNDERFITTING (model does not fit well the training data)
  - Measure of error of the model comparing predicted output against actual output of training data

    - Linear functions typically have high bias, usually does not fit training data accurately
- Variance : how accurately we can fit the a new data set in this function
    - High Variance = OVERFITTING (can't generalize, model only fits the training data well)
    - Measure of error of the model comparing predicted output against actual output of testing data
    - High-degree polynomials typically have high variance, usually they fit training data too closely

d. Decision tree classifier

- For a supervised learning problem with a finite set of expected outputs.
- It performs a sequence of tests, starting from the root, to reach a decision at the leaf. It adopts a greedy divide-and-conquer strategy, testing the most important attribute first, then recursively solving the subproblems. The attribute with highest importance should be the root.
- Use information gain defined in terms of entropy to measure the importance. An attribute with low information gain is irrelevant.
- Entropy is the measure of randomness/uncertainty. The more information (certainty), the less entropy.

Entropy: $H(V) \;=\; \sum_k \; P(Vk)\, log2\,(\,1/P(Vk)) \;=\; \sum_k \; P(Vk)\, log2\;\, 1 \;-\; P(Vk)\, log2\;\, P(Vk)$

Since $\log_2 1 = 0$, $H(V) \;=\; -\, \sum_k \; P(Vk)\, log2\;\, P(Vk)$

Information Gain: ΔH = H(dataset) - H(attribute)

Continuous attribute values can be categorized in ranges.

Advantages:

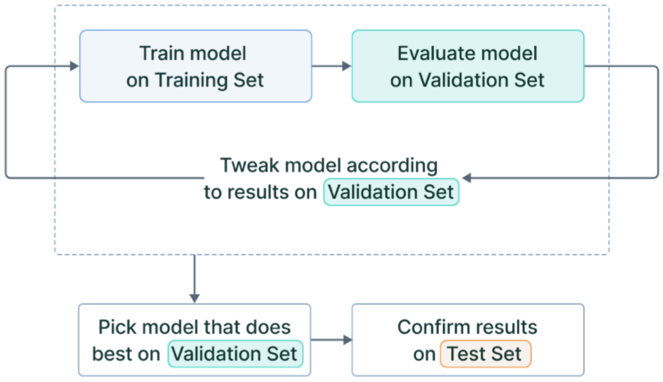
- Easy to understand
- Scalability to large sets
- Handle both discrete and continuous values

Disadvantages:

- If a tree is too deep (many features) it is computational expensive (runtime)
- Unstable, any feature addition changes the whole tree.
- Data must be representative of what you're trying to predict, changes in the training dataset change the decision tree

e. Need for validation dataset

- A validation dataset (or development set, or dev set) can be used to help fine tune the algorithm during training, before testing it.
- The validation dataset is used to fine tune parameters, select features, and make decisions regarding the learning algorithm.
    - First, split the dataset into training and testing, or have a completely separated testing dataset.
    - Then, split the training dataset into training and validation. Depending on its size, use k-fold cross-validation during training. The k-fold cross-validation is a way of separating the training dataset and the validation dataset when we have a limited dataset, but to have a k-fold leads to k iterations, increasing the run time of the algorithm (computational expensive).
    - It's important to ensure that the training and validation dataset are representative of the future data the model will need to evaluate.
- The most important is to have a completely separate testing dataset to evaluate the algorithm (the testing dataset cannot be seen by the model during training in any way).
- Training dataset: used to fit the model.
- Validation dataset: used to validate the model during training.
- Testing dataset: used to evaluate model accuracy.
- Good model:
    - Consistent
    - Generalized
    - High accuracy
    - Low bias and low variance (but they are inversely related)


Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

- Steps for choosing a good model:
    - Understand dataset
    - Choose representative training, validation, and testing datasets
    - Try different approaches to fine tune the model using the validation dataset
    - Chose the model that works best, highest accuracy
    - Use the testing dataset to evaluate the model