Decision Tree

Gender	Occupation	Suggestion
F	Student	PUBG
F	Programmer	Github
M	Programmer	Whatsapp
F	Programmer	Github
M	Student	PUBG
M	Student	PUBG

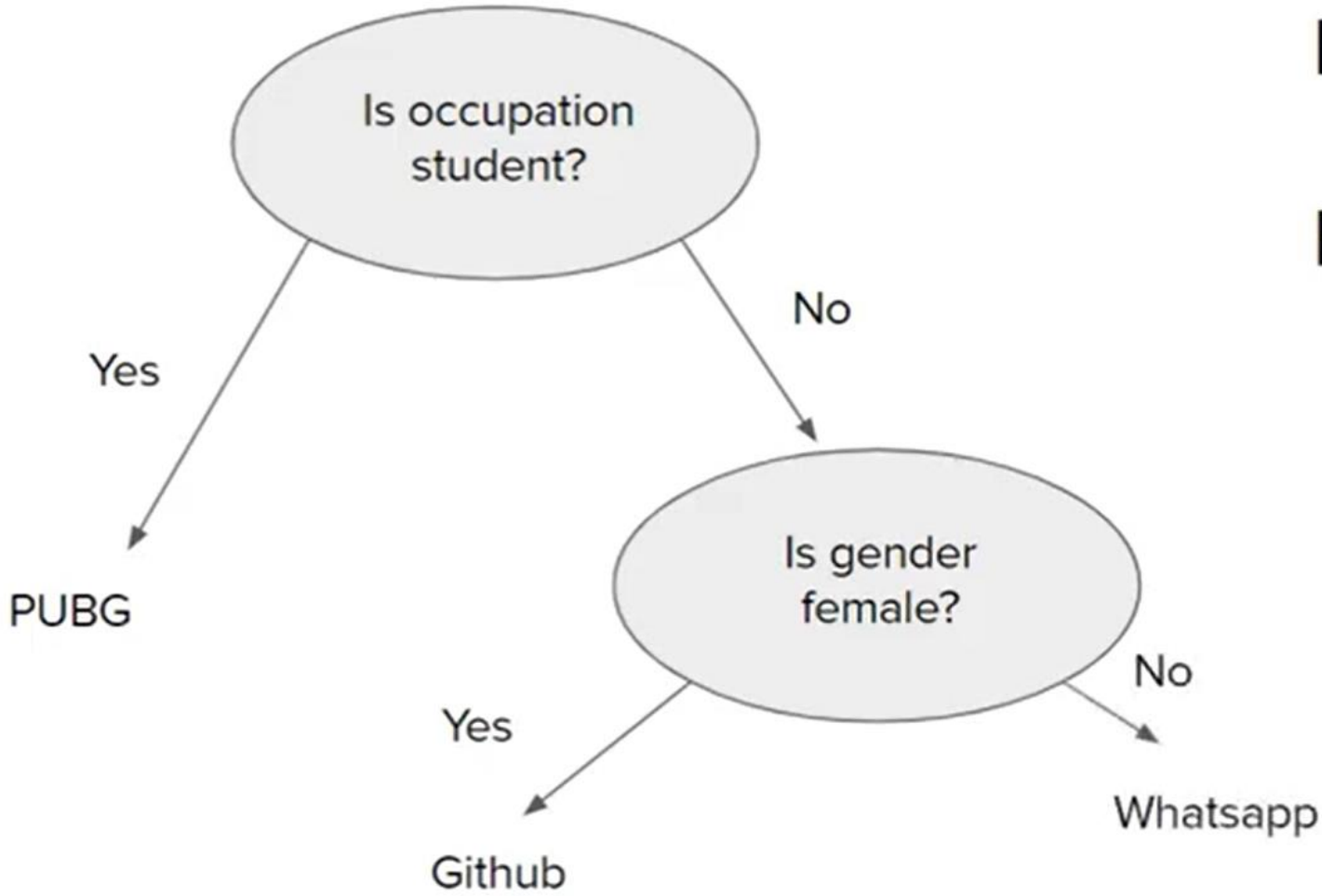
Gender	Occupation	Suggestion
F	Student	PUBG
F	Programmer	Github
M	Programmer	Whatsapp
F	Programmer	Github
M	Student	PUBG
M	Student	PUBG

```
If occupation==student
    print(PUBG)
Else
    If gender==female
        print(Github)
    Else
        print(Whatsapp)
```

Where is the Tree?

```
If occupation==student
    print(PUBG)
Else
    If gender==female
        print(Github)
    Else
        print(Whatsapp)
```

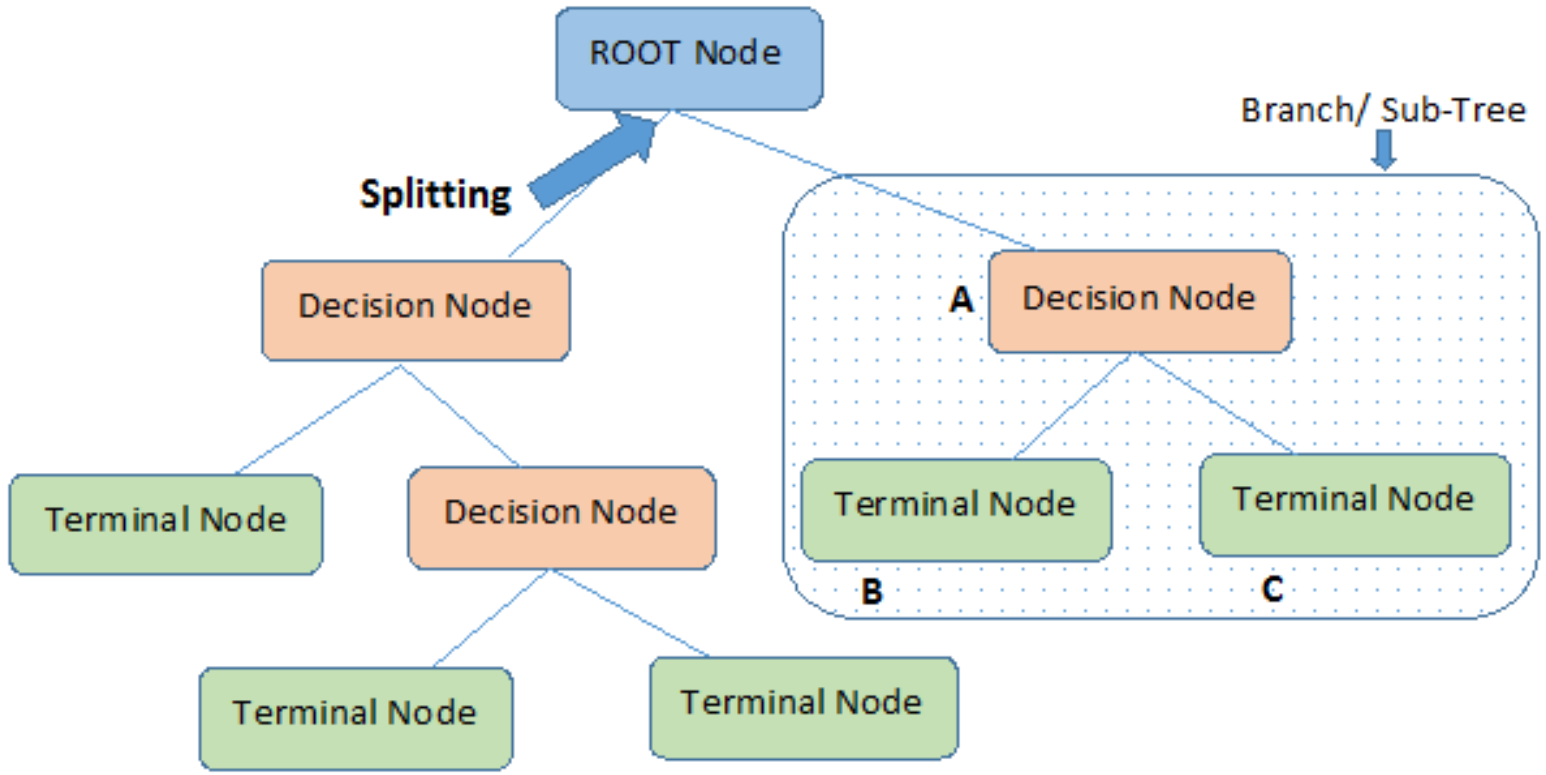
Where is the Tree?

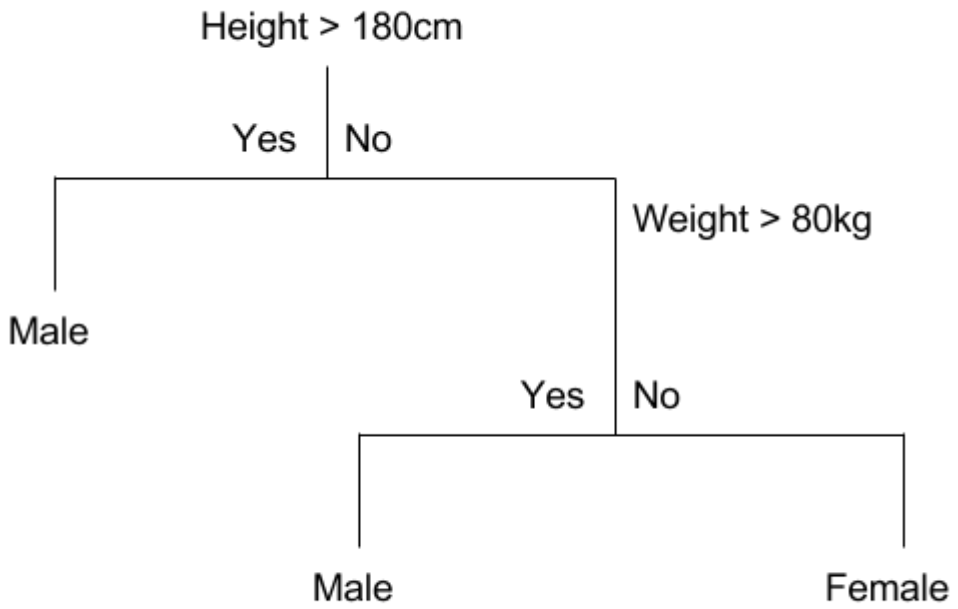



```
If occupation==student
    print(PUBG)
Else
    If gender==female
        print(Github)
    Else
        print(Whatsapp)
```

Decision Tree

- classification and prediction tool having a tree-like structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
- used for both classification and regression problems


ROOT Node
Branch/ Sub-Tree
Splitting
Decision Node
A
Decision Node
Terminal Node
Decision Node
Terminal Node
Terminal Node
B
C
Terminal Node
Terminal Node



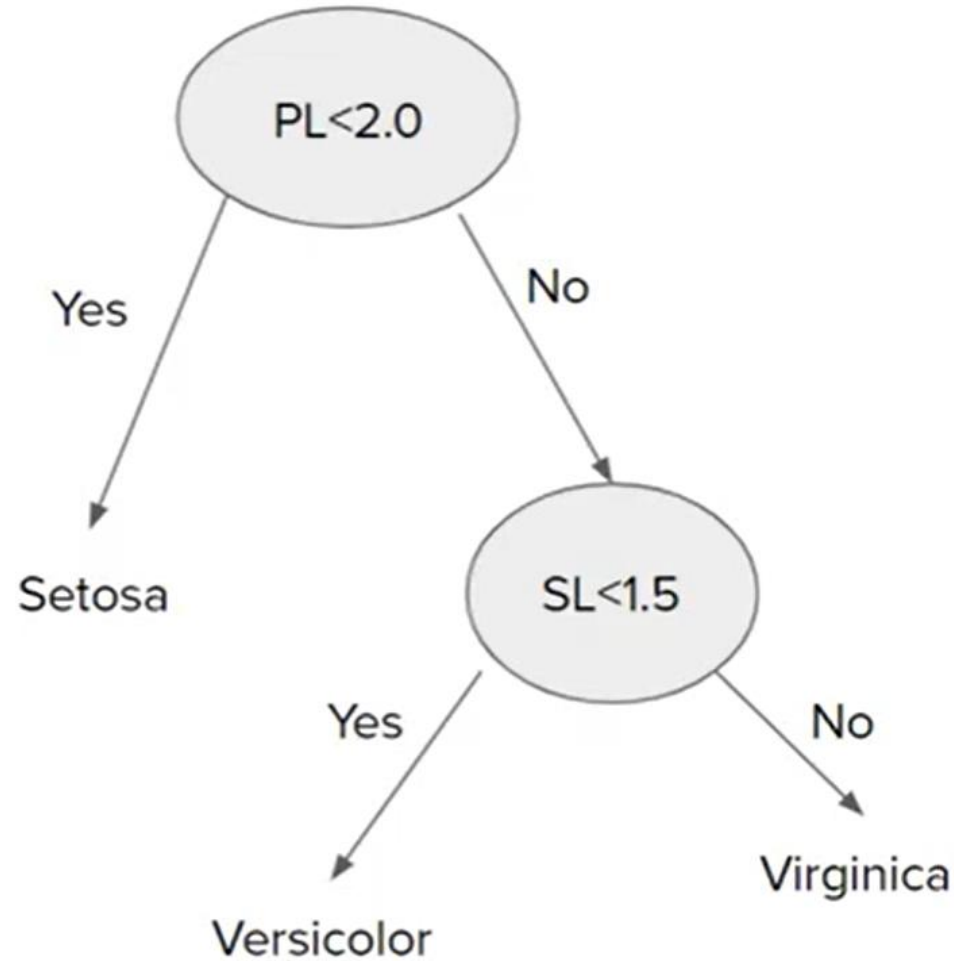
- Here If Height > 180cm or if height < 180cm and weight > 80kg person is male. Otherwise female.

What if we have numerical data?

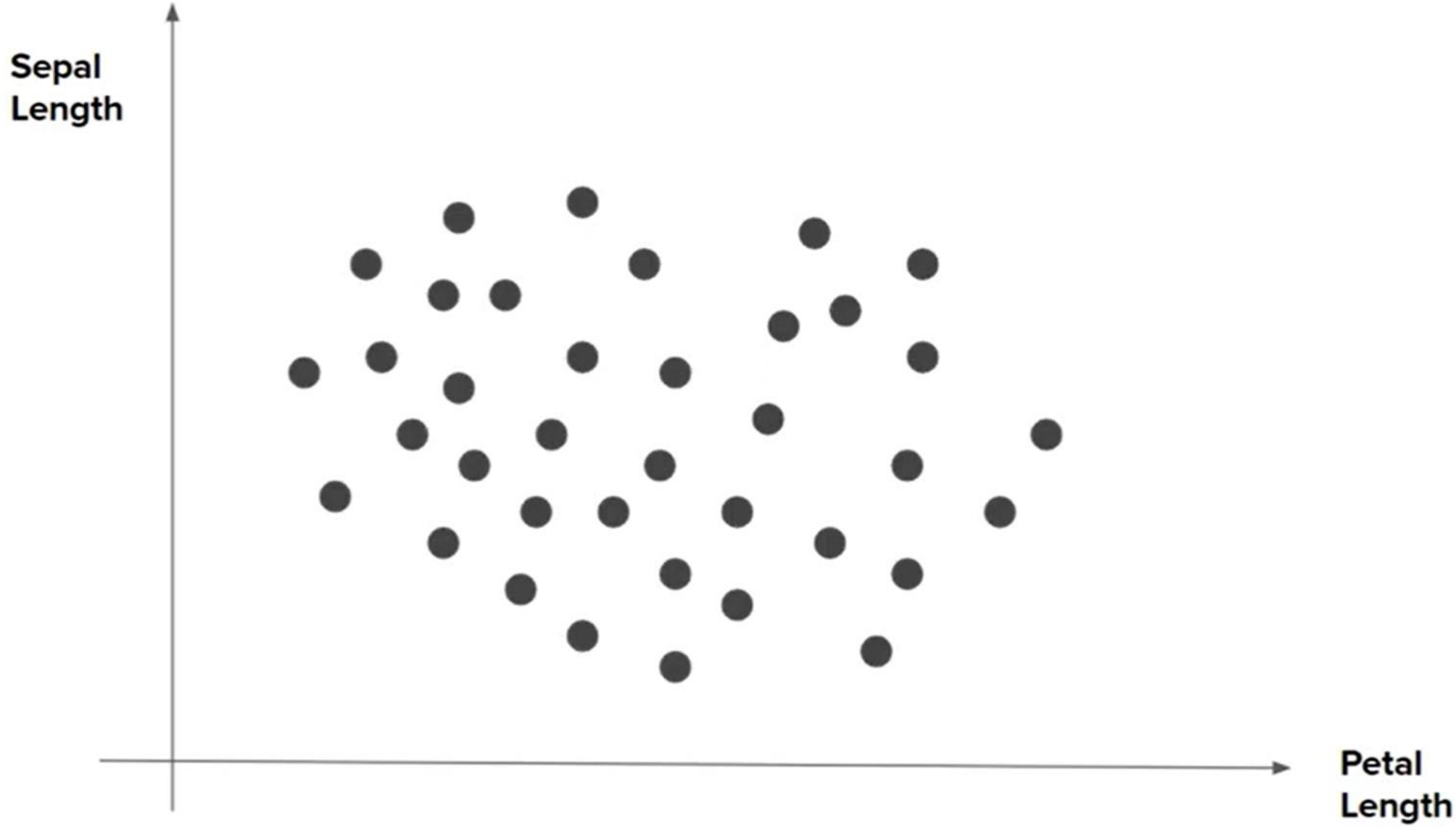
Petal Length	Sepal Length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.88	Setosa

What if we have numerical data?

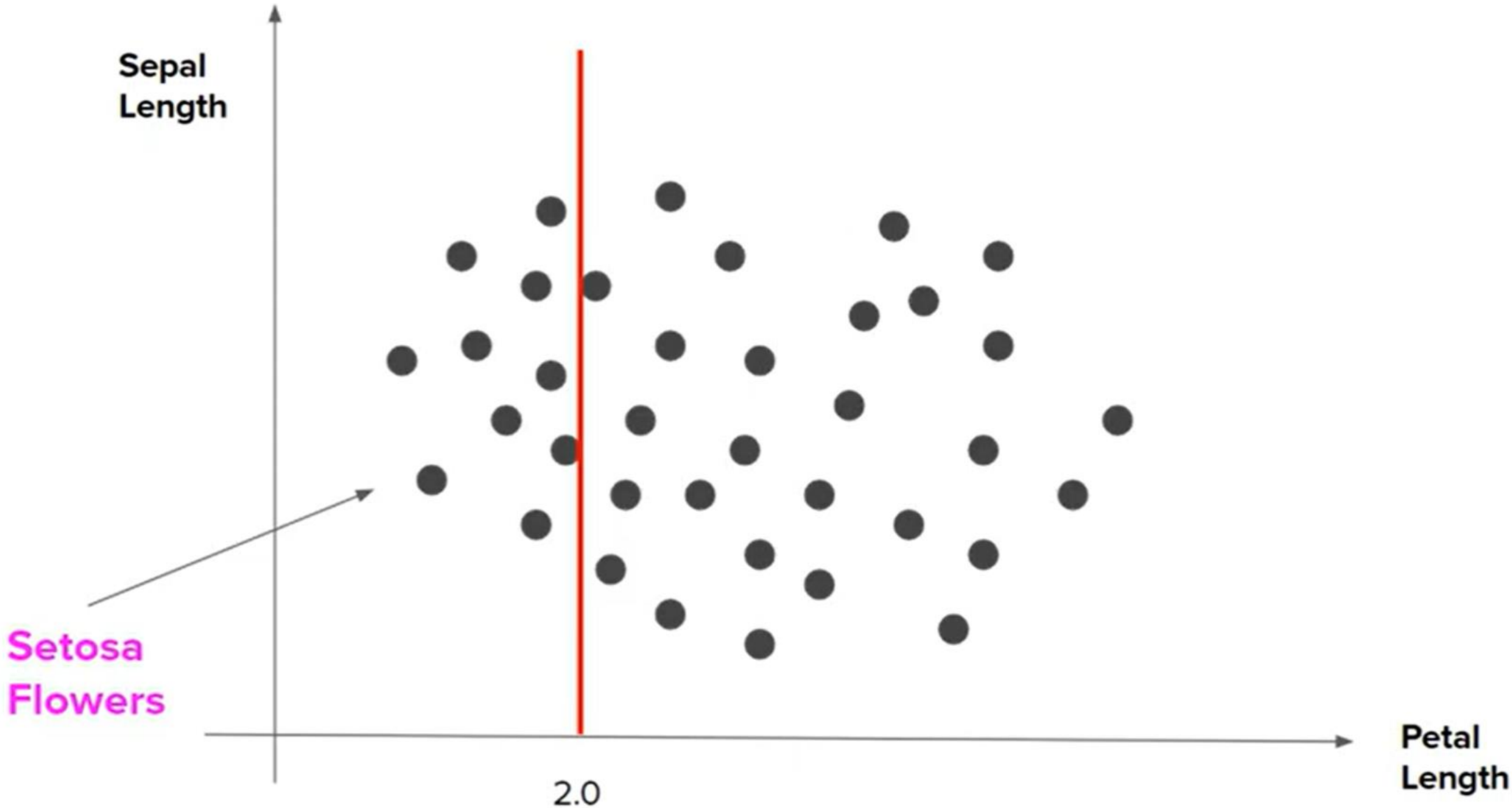
Petal Length	Sepal Length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.88	Setosa



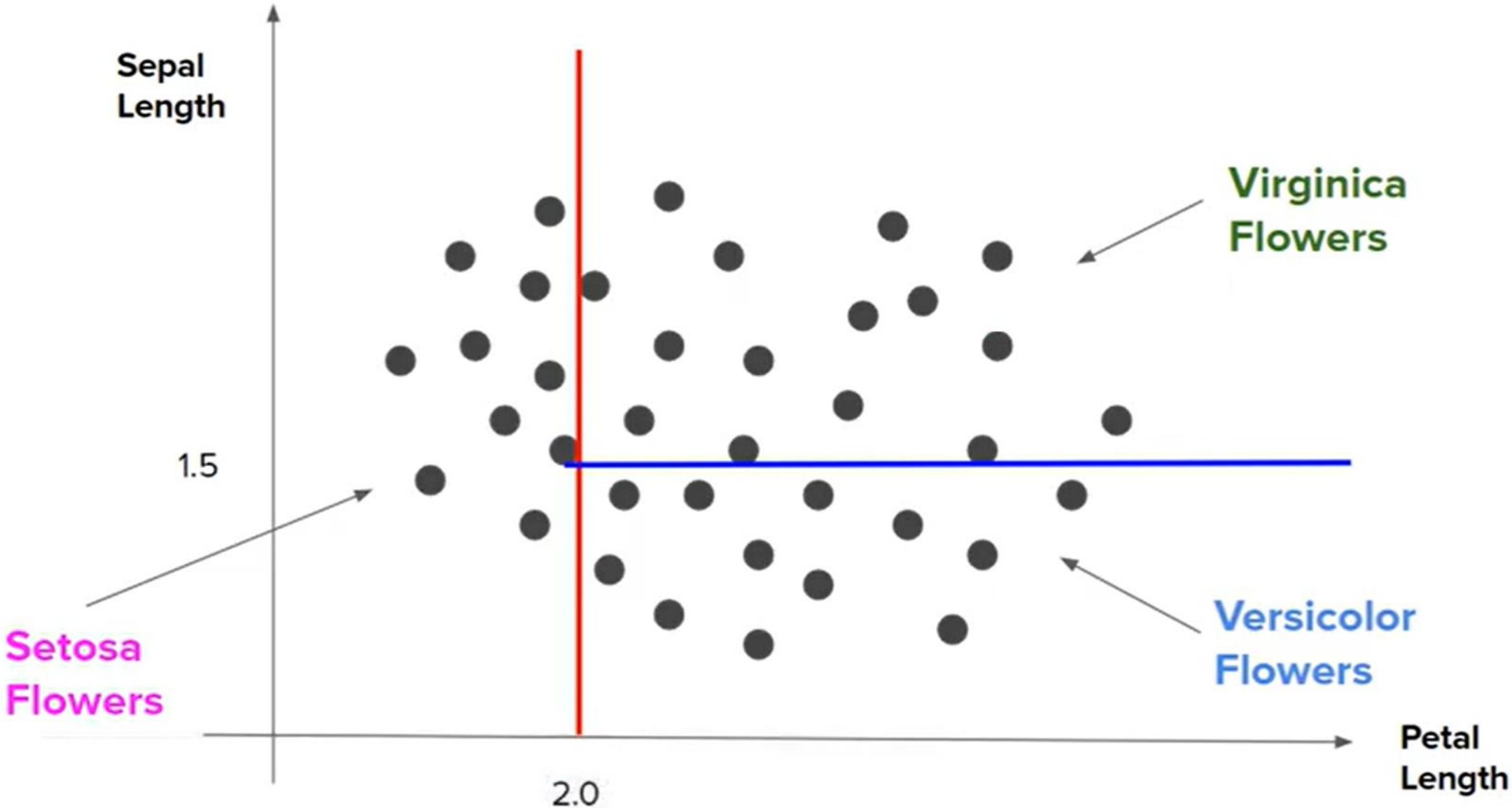
Geometric Intuition



Geometric Intuition



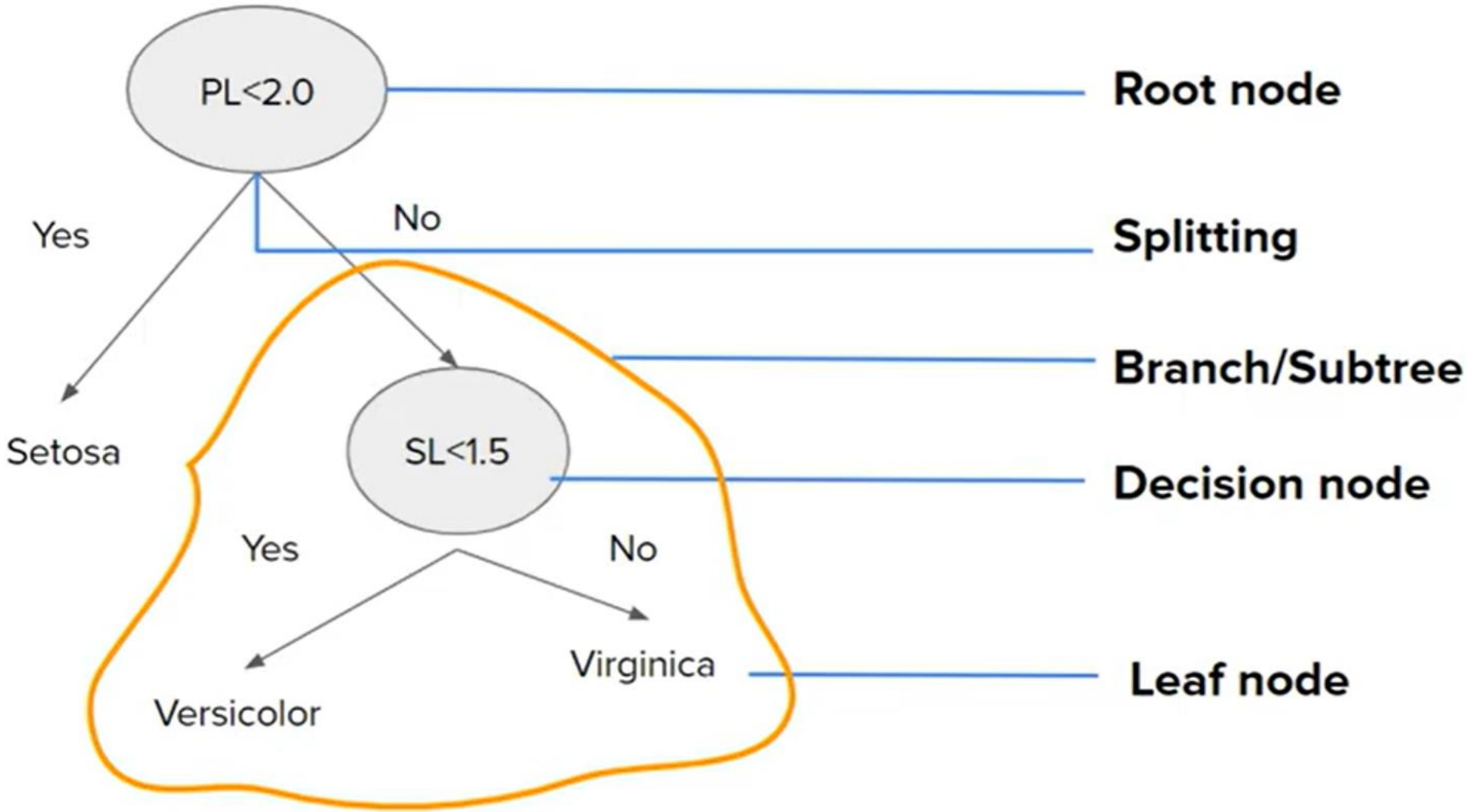
Geometric Intuition



Pseudo code

- Begin with your training dataset, which should have some feature variables and classification or regression output.
- Determine the “best feature” in the dataset to split the data on; more on how we define “best feature” later
- Split the data into subsets that contain the correct values for this best feature. This splitting basically defines a node on the tree i.e each node is a splitting point based on a certain feature from our data.
- Recursively generate new tree nodes by using the subset of data created from step 3.

Terminology



Some unanswered questions

How to decide which column should be considered as root node?

How to select subsequent decision nodes?

How to decide splitting criteria in case of numerical columns?

Entropy

- Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

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

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

$$H(d) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$$

$$H(d) = 0.97$$

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

$$H(d)= -P_y \log_2(P_y) -P_n \log_2(P_n)$$

$$H(d)= -2/5 \log_2(2/5) -3/5 \log_2(3/5)$$

$$H(d)= 0.97$$

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$$H(d)= -P_y \log_2(P_y) -P_n \log_2(P_n)$$

$$H(d)= -1/5 \log_2(1/5) -4/5 \log_2(4/5)$$

$$H(d)= 0.72$$

Salary	Age	Purchase
20000	21	No
10000	45	No
60000	27	No
15000	31	No
12000	18	No

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

$$H(d) = -0/5 \log_2(0/5) - 5/5 \log_2(5/5)$$

$$H(d) = 0$$

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
30000	30	Maybe
12000	18	No
40000	40	Maybe
20000	20	Maybe

$$H(d) = - P_y \log_2(P_y) - P_n \log_2(P_n) - P_m \log_2(P_m)$$

$$H(d) = -2/8 \log_2(2/8) \; -3/8 \log_2(3/8) - 3/8 \log_2(3/8)$$

$$H(d) = 1.56$$

Observation

- More the uncertainty more is entropy
- For a 2 class problem the min entropy is 0 and the max is 1
- For more than 2 classes the min entropy is 0 but the max can be greater than 1
- Both $\log_2$ or $\log_e$ can be used to calculate entropy

Information Gain

Information Gain, is a metric used to train Decision Trees. Specifically, this metric measures the quality of a split.

The information gain is based on the decrease in entropy after a data-set is split on an attribute.Constructing a decision tree is all about finding attribute that returns the highest information gain

Information Gain = E(Parent) - {Weighted Average} * E{Children}

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Step 1:

Entropy of Parent

$E(P) = -p_y \log_2(p_y) - p_n \log_2(p_n)$

$= 9/14 \log_2(9/14) - 5/14/\log_2(5/14)$

$E(P) = \mathbf{0.94}$

Information Gain

- amount of information gained about a random variable or signal from observing another random variable.
- It can be considered as the difference between the entropy of parent node and weighted average entropy of child nodes.

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

- There are many algorithms there to build a decision tree. They are

1. **CART** (Classification and Regression Trees) — This makes use of Gini impurity as the metric.
2. **ID3** (Iterative Dichotomiser 3) — This uses entropy and information gain as metric.

Classification using the ID3 algorithm

- Consider whether a dataset based on which we will determine whether to play football or not.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

- As the first step, we have to find the parent node for our decision tree. For that follow the steps:

1. ***Find the entropy of the class variable.***

$$E(S) = -[(9/14)\log(9/14) + (5/14)\log(5/14)] = 0.94$$

note: Here typically we will take log to base 2.Here total there are 14 yes/no. Out of which 9 yes and 5 no.Based on it we calculated probability above.

- From the above data for outlook we can arrive at the following table

		play		
		yes	no	total
	sunny	3	2	5
Outlook	overcast	4	0	4
	rainy	2	3	5
				14

2. calculate average weighted entropy.

ie, we have found the total of weights of each feature multiplied by probabilities.

- E(S, outlook) = (5/14)*E(3,2) + (4/14)*E(4,0) + (5/14)*E(2,3)

  = (5/14)(-(3/5)log(3/5)-(2/5)log(2/5))+ (4/14)(0) + (5/14)((2/5)log(2/5)-(3/5)log(3/5))

  = 0.693

3. Find the information gain.

It is the difference between parent entropy and average weighted entropy we found above.

- IG(S, outlook) = 0.94 - 0.693 = 0.247

4. Similarly find Information gain for Temperature, Humidity, and Windy.

- IG(S, Temperature) = 0.940 - 0.911 = 0.029
- IG(S, Humidity) = 0.940 - 0.788 = 0.152
- IG(S, Windy) = 0.940 - 0.8932 = 0.048
- ***Now select the feature having the largest information gain***. Here it is Outlook. So it forms the **first node(root node)** of our decision tree.

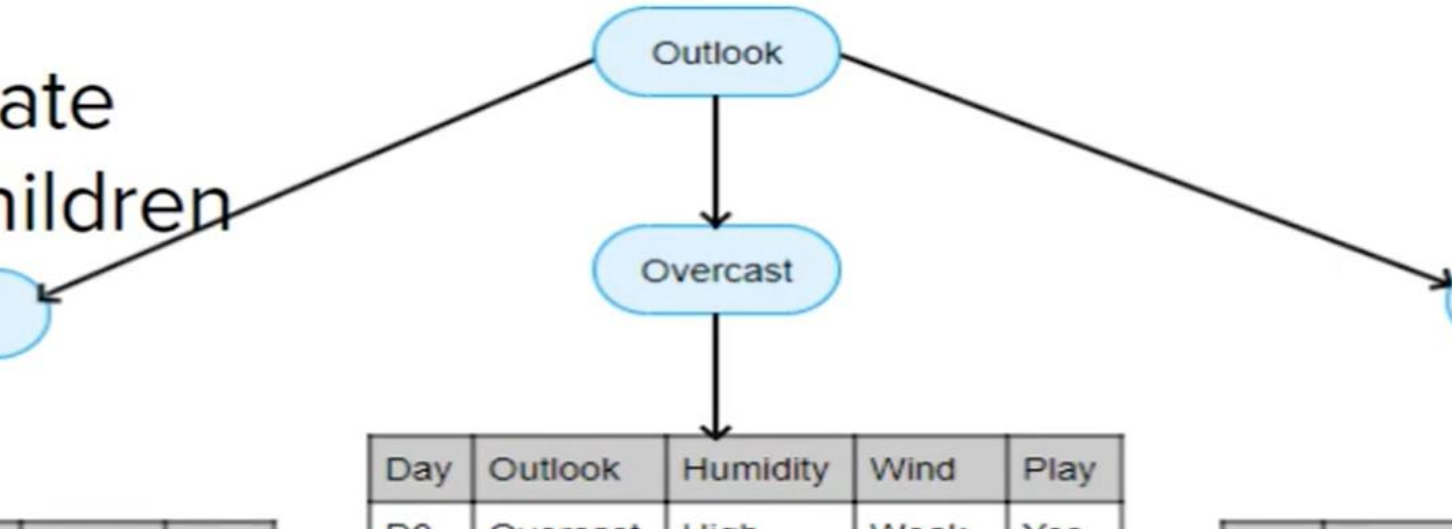
Now our data look as follows

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Overcast	Hot	High	Weak	Yes
Overcast	Cool	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Rain	Mild	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Calculate Entropy for Children

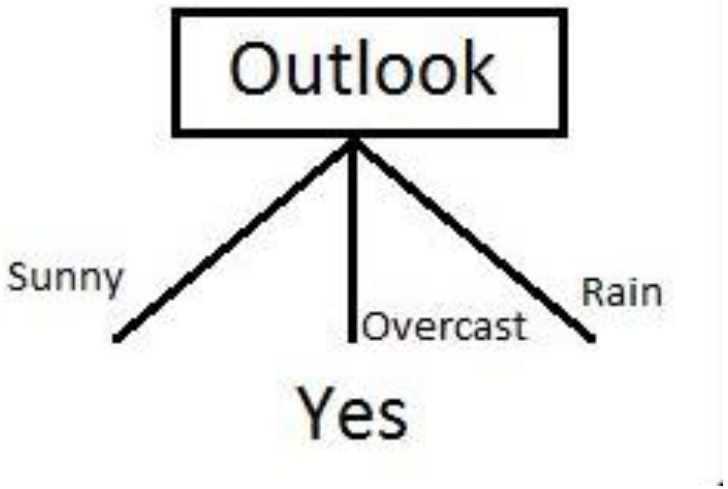


Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes

Day	Outlook	Humidity	Wind	Play
D3	Overcast	High	Weak	Yes
D7	Overcast	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes

Day	Outlook	Humidity	Wind	Play
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D10	Rain	Normal	Weak	Yes
D14	Rain	High	Strong	No

- Since overcast contains only examples of class ‘Yes’ we can set it as yes. That means If outlook is overcast football will be played. Now our decision tree looks as follows.



- The next step is to find the next node in our decision tree.
- Now we will find one under sunny. We have to determine which of the following Temperature, Humidity or Wind has higher information gain.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

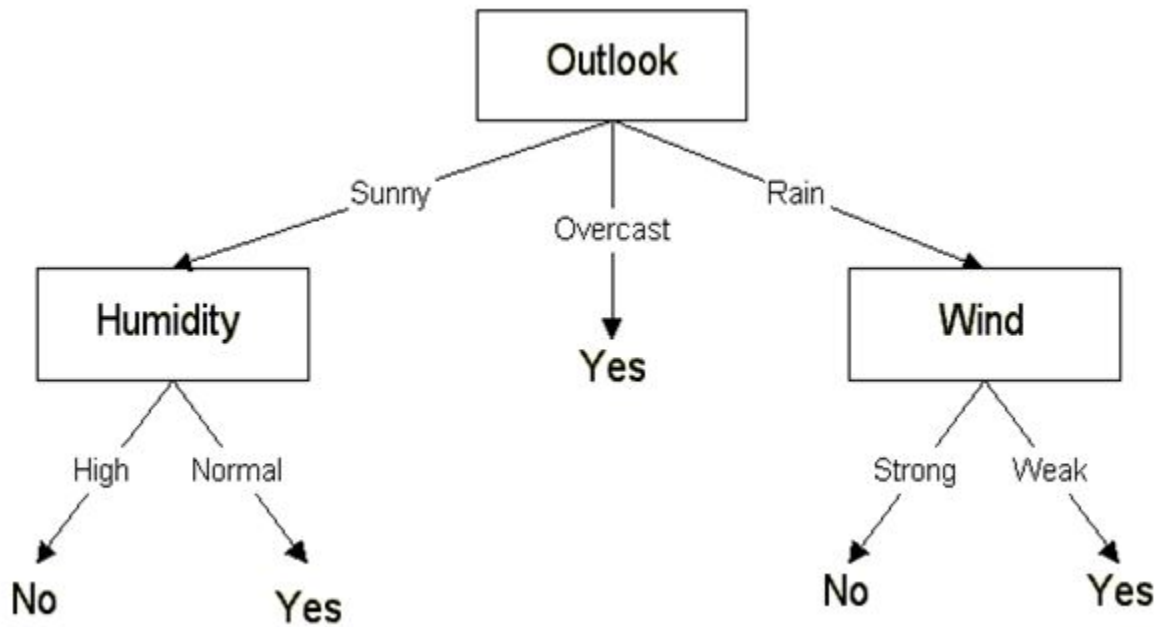
- Calculate parent entropy E(sunny)
- E(sunny) = (-(3/5)log(3/5)-(2/5)log(2/5)) = 0.971.
- Now Calculate the information gain of Temperature. IG(sunny, Temperature)

		play		
		yes	no	total
	hot	0	2	2
Temperature	cool	1	1	2
	mild	1	0	1
				5

- E(sunny, Temperature) = (2/5)*E(0,2) + (2/5)*E(1,1) + (1/5)*E(1,0)=2/5=0.4
- Now calculate information gain.
  - IG(sunny, Temperature) = 0.971–0.4 =0.571
- Similarly we get
  - IG(sunny, Humidity) = 0.971
  - IG(sunny, Windy) = 0.020
- Here IG(sunny, Humidity) is the largest value. So Humidity is the node that comes under sunny.

	play	
Humidity	yes	no
high	0	3
normal	2	0

- For humidity from the above table, we can say that play will occur if humidity is normal and will not occur if it is high. Similarly, find the nodes under rainy.
- ***Note: A branch with entropy more than 0 needs further splitting.***
- Finally, our decision tree will look as below:


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Gini Impurity

- Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$G = 1 - (Py^2 + Pn^2)$

$G = 1 - (4/25 + 9/25)$

$G = 0.48$

$G = 1 - (Py^2 + Pn^2)$

$G = 1 - (1/25 + 16/25)$

$G = 0.32$

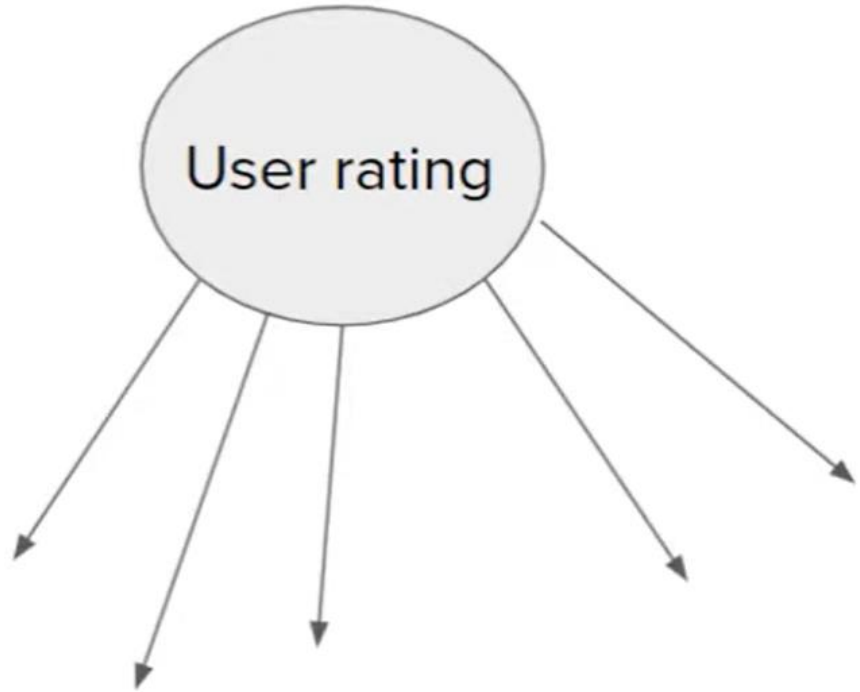
- Entropy-slower, but gives a more balanced tree
- gini-faster

Handling numerical data

S No	User Rating	Downloaded
1	3.5	Yes
2	4.6	Yes
3	2.2	No
4	1.6	Yes
5	4.1	No
6	3.9	No
7	3.2	No
9	2.9	Yes
10	4.8	Yes
11	3.3	No
12	2.5	Yes
13	1.9	Yes

S No	User Rating	Downloaded
1	3.5	Yes
2	4.6	Yes
3	2.2	No
4	1.6	Yes
5	4.1	No
6	3.9	No
7	3.2	No
9	2.9	Yes
10	4.8	Yes
11	3.3	No
12	2.5	Yes
13	1.9	Yes

data['user_rating'].nunique()=n



We will then have n subtrees

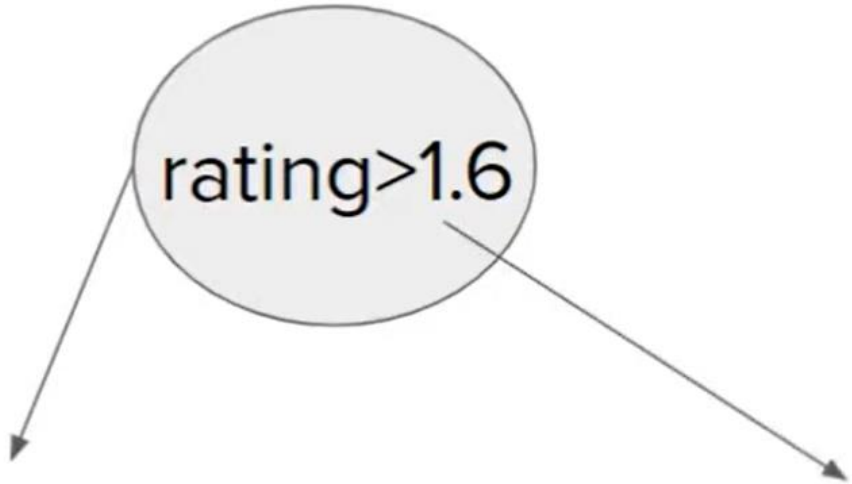
Step 1:

Sort the data on the basis of numerical column

S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

Step 2:

Split the entire data on the basis of every value of user_rating

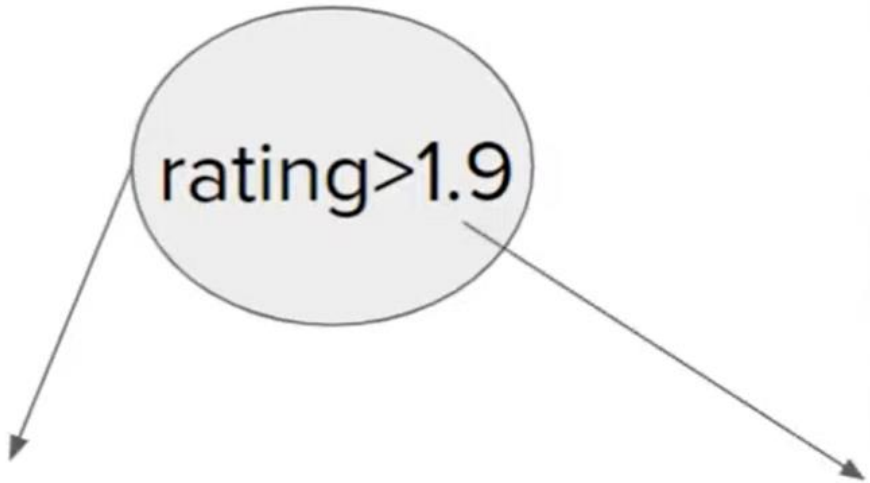


S No	User Rating	Downloaded
1	1.6	Yes

S No	User Rating	Downloaded
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

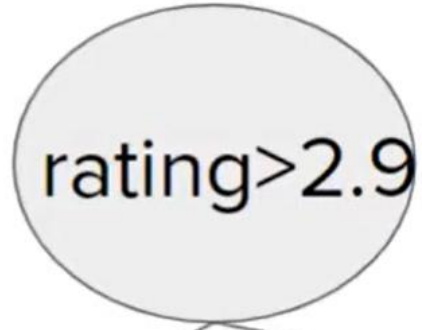
Step 2:

Split the entire data on the basis of every value of user_rating



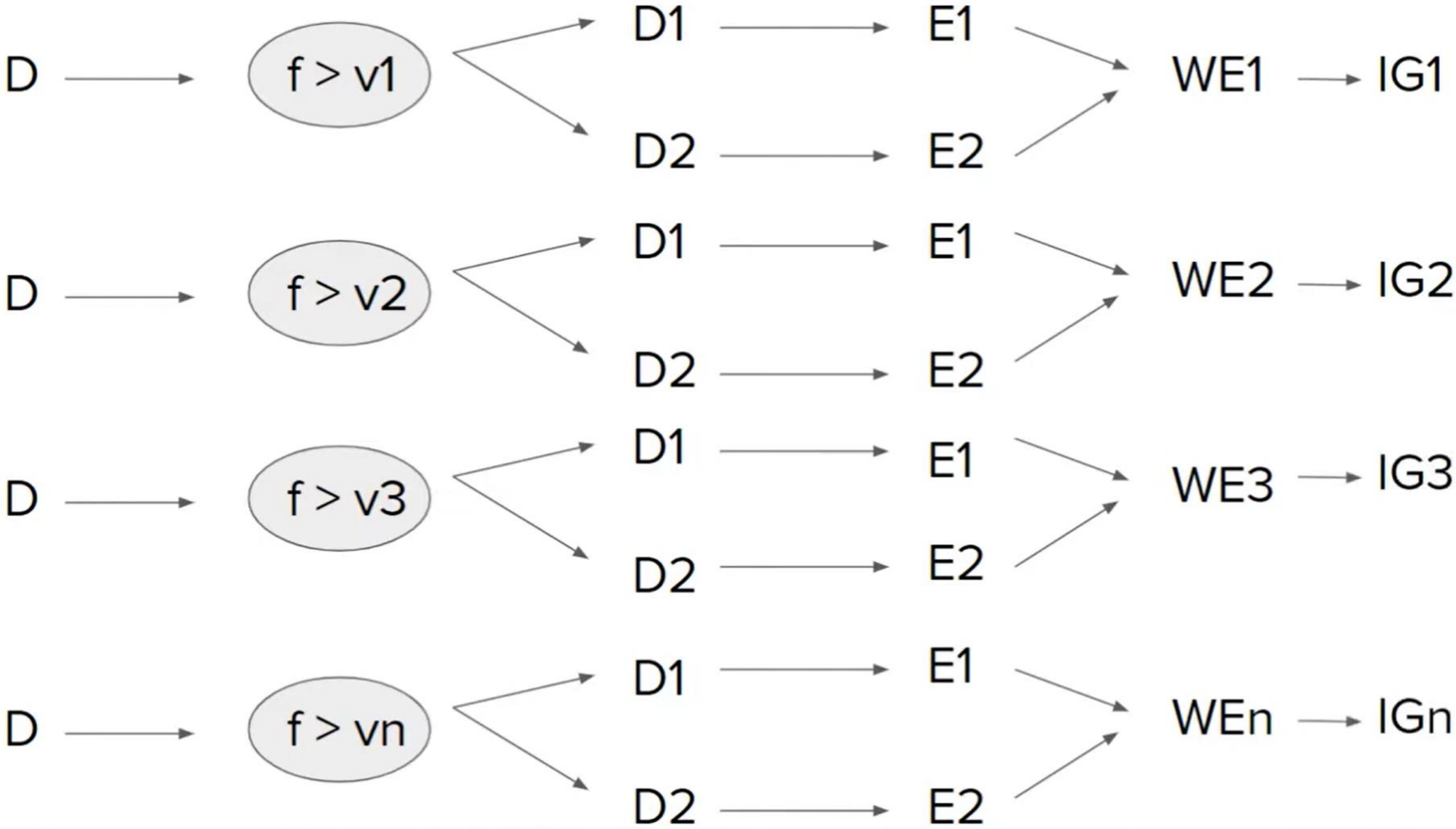
S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes

S No	User Rating	Downloaded
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes



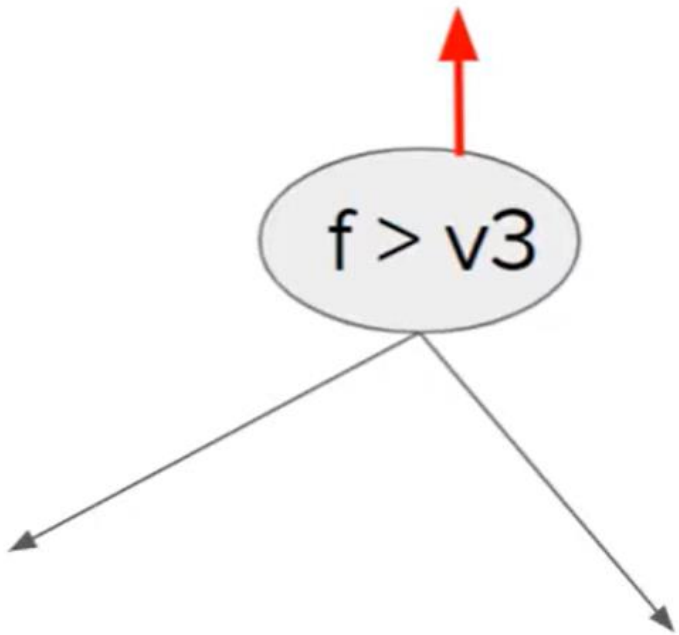
S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes

S No	User Rating	Downloaded
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes


D
f > v1
D1
D2
E1
E2
WE1
IG1
D
f > v2
D1
D2
E1
E2
WE2
IG2
D
f > v3
D1
D2
E1
E2
WE3
IG3
D
f > vn
D1
D2
E1
E2
WEn
IGn

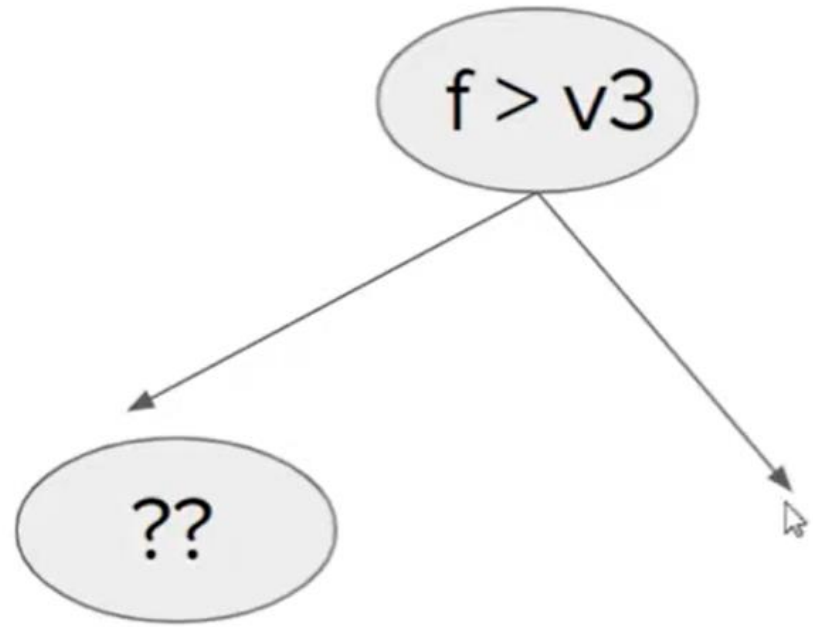
Step 5:

Max{ IG1, IG2, IG3....., IGn }

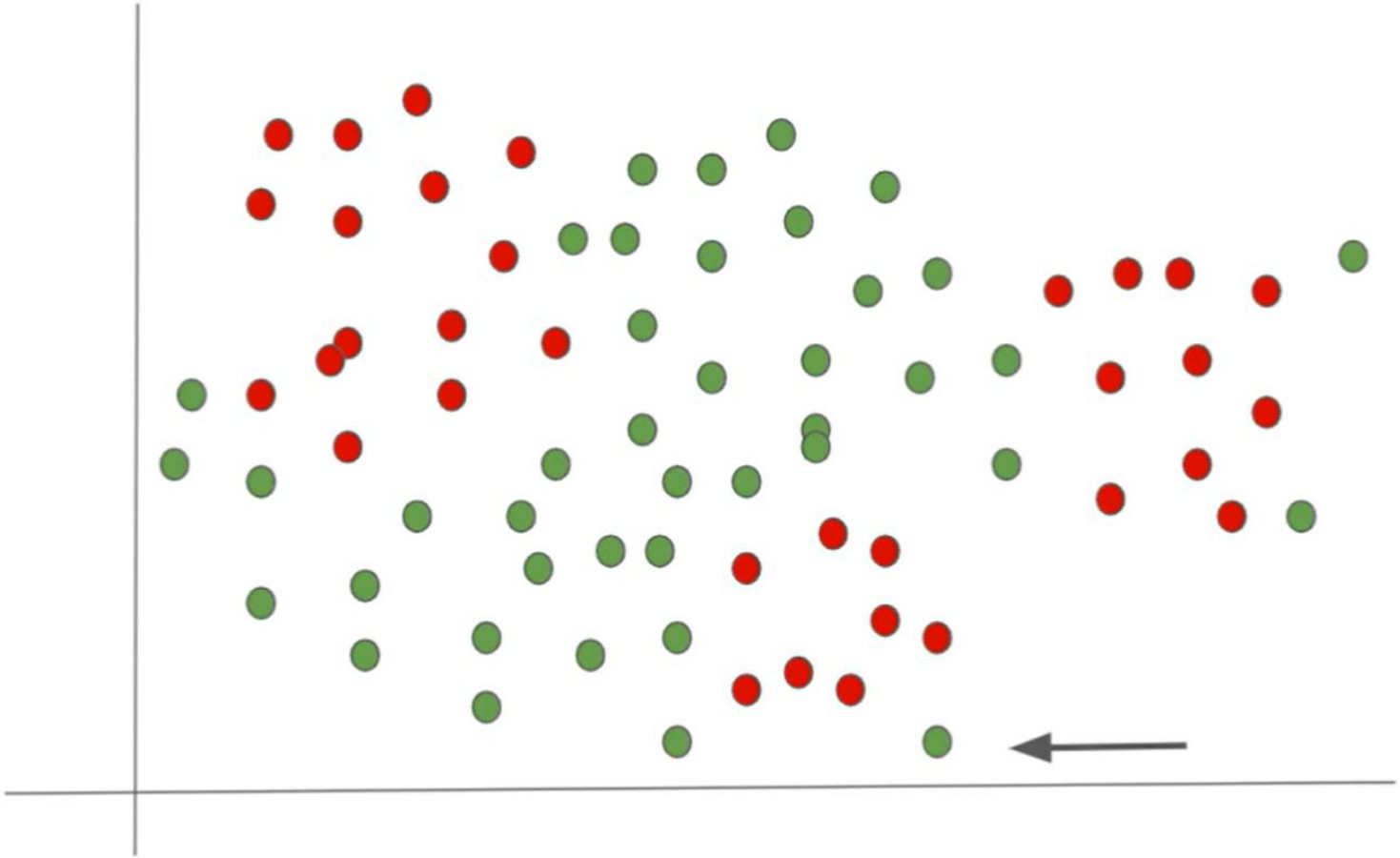


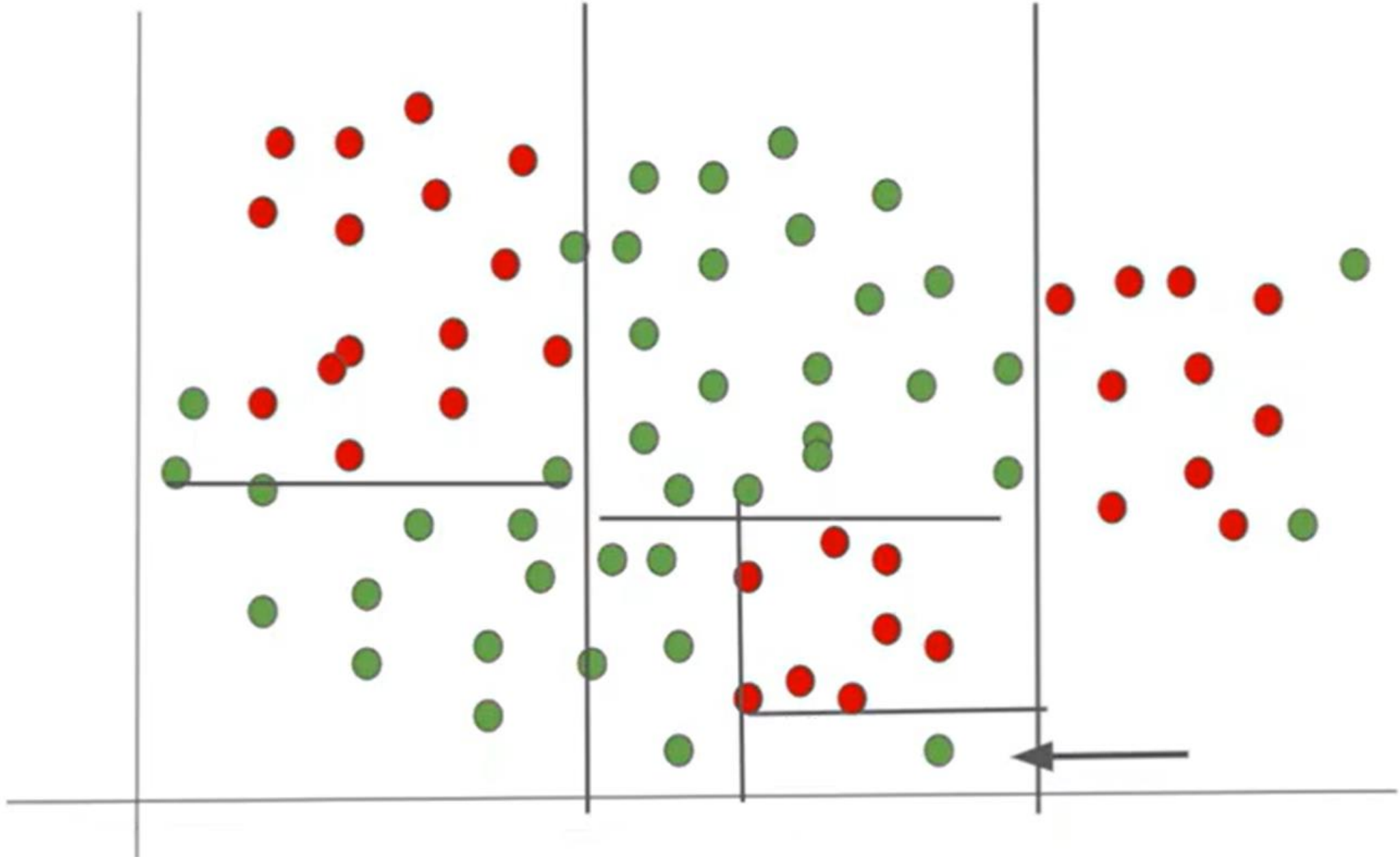
Step 6:

Do this recursively until you get all the leaf nodes



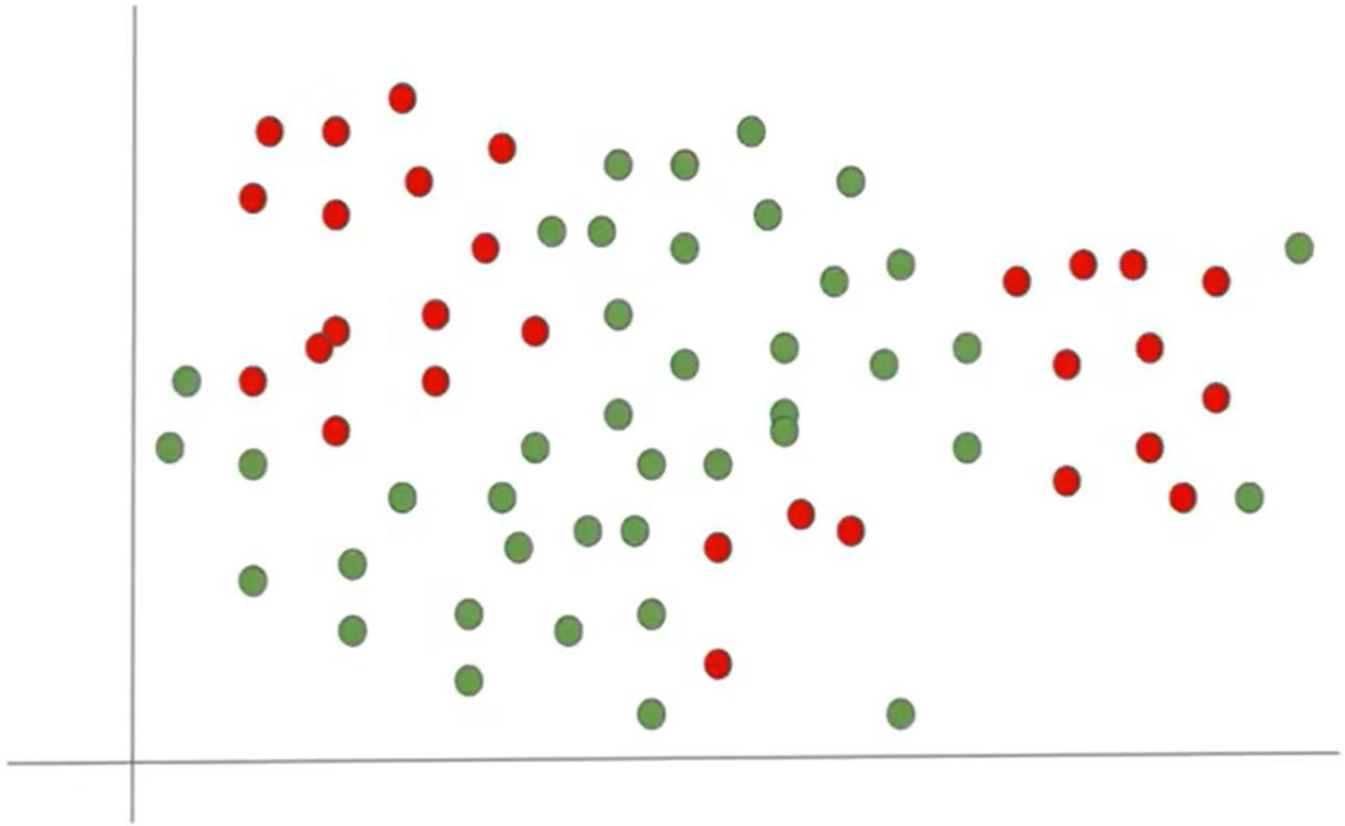
Overfitting in decision tree

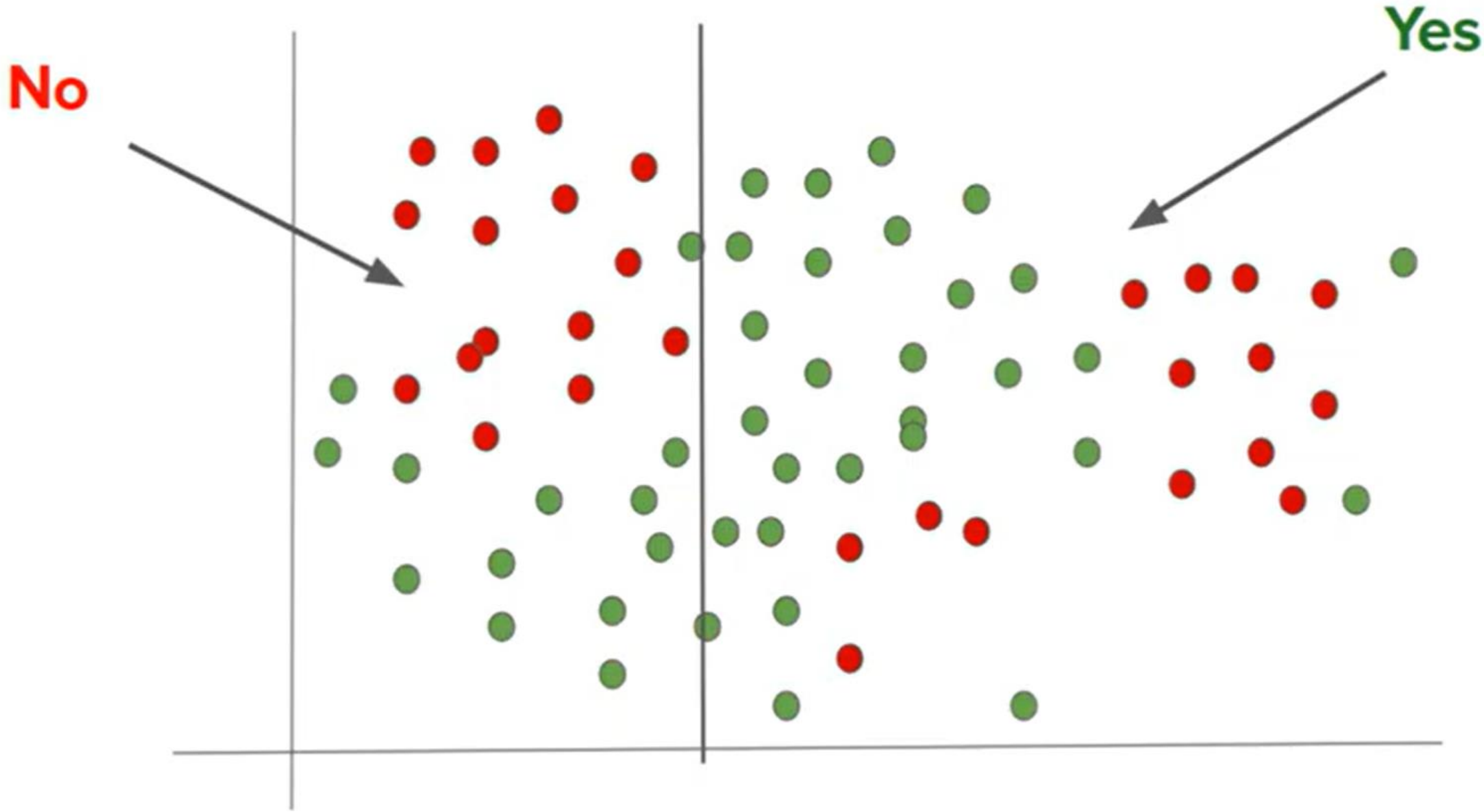




- Overfitting in decision trees can occur when the tree is too complex and captures noise in the training data instead of the underlying patterns. This can happen in several ways:
  - Tree depth: A decision tree can overfit when it is too deep. This means that the tree is too complex and captures noise in the training data instead of the underlying patterns.
  - Small training set: If the training set is too small, a decision tree may overfit by capturing noise in the limited data rather than the underlying patterns.
  - Irrelevant features: If the decision tree uses irrelevant features, it may overfit by capturing noise in those features rather than the underlying patterns.

Underfitting in decision tree




No
Yes

- Underfitting in decision trees can occur when the tree is too simple and fails to capture the underlying patterns in the data. This can happen in several ways:
  - Tree depth: If the decision tree is too shallow, it may not capture the underlying patterns in the data.
  - Lack of features: If the decision tree does not have access to enough relevant features, it may not be able to capture the underlying patterns in the data.
  - Insufficient training data: If the decision tree does not have enough training data, it may not be able to capture the underlying patterns in the data.
- To avoid underfitting in decision trees, it is important to use techniques such as increasing the depth of the tree, providing more relevant features, and increasing the size of the training set. However, it is also important to avoid overfitting, so finding the right balance between model complexity and performance is essential.

How to avoid overfitting?

1. Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
2. Allow the tree to *overfit* the data, and then *post-prune* the tree
   - Split the training in two parts (training and validation) and use validation to assess the utility of *post-pruning*
     - Reduced error pruning
     - Rule Post pruning

Reduced-error pruning

1. Each node is a candidate for pruning
2. *Pruning* consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification
3. Nodes are removed only if the resulting tree performs better on the **validation set.**
4. Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned.
5. Pruning stops when no pruning increases accuracy

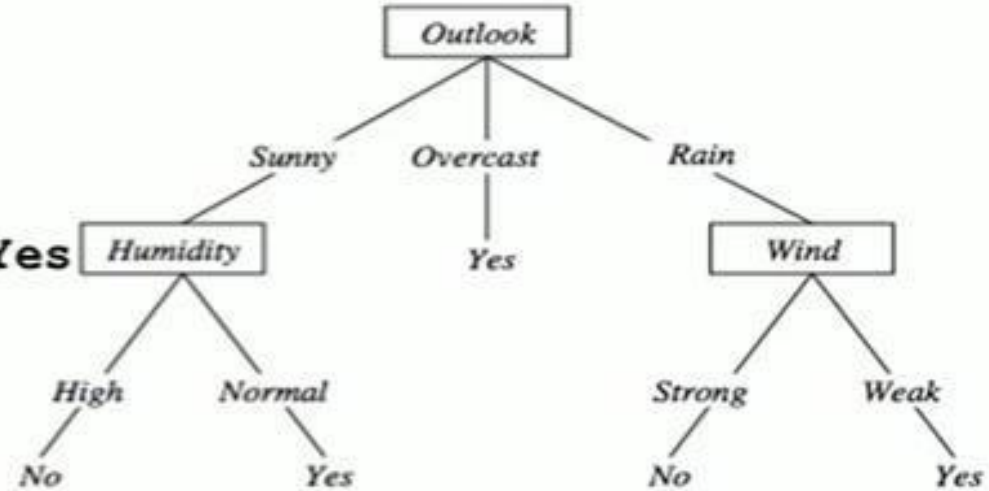
Reduced-error pruning

Rule post-pruning

1. Create the decision tree from the training set
2. Convert the tree into an equivalent set of rules
   - Each path corresponds to a rule
   - Each node along a path corresponds to a pre-condition
   - Each leaf classification to the post- condition
3. Prune (generalize) each rule by removing those preconditions whose removal improves accuracy over validation set

Rule post-pruning

1. `Outlook=sunny ^ humidity=high -> No`
2. `Outlook=sunny ^ humidity=normal -> Yes`
3. `Outlook=overcast -> Yes`
4. `Outlook=rain ^ wind=strong -> No`
5. `Outlook=rain ^ wind=weak -> Yes`



Compare first rule to:

Outlook=sunny->No

Humidity=high->No

Calculate accuracy of 3 rules based on validation set and pick best version.

Rule post-pruning

Why converting to rules?

1. Each distinct path produces a different rule: a condition removal may be based on a local (contextual) criterion. Node pruning is global and affects all the rules
2. In rule form, tests are not ordered and there is no book-keeping involved when conditions (nodes) are removed
3. Converting to rules improves readability for humans

Advantages

Intuitive and easy to understand

Minimal data preparation is required

The cost of using the tree for inference is **logarithmic** in the number of data points used to train the tree

Disadvantages

Overfitting

Prone to errors for imbalanced datasets