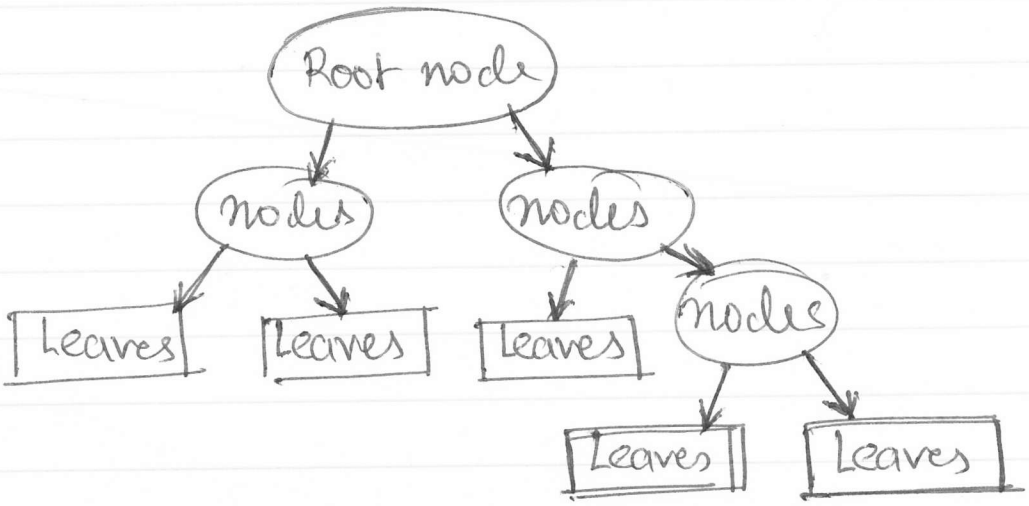
Decision Tree

— Most commonly used for classification based on Categorical or Numerical data that can be discrete or Continious or a Combination

—



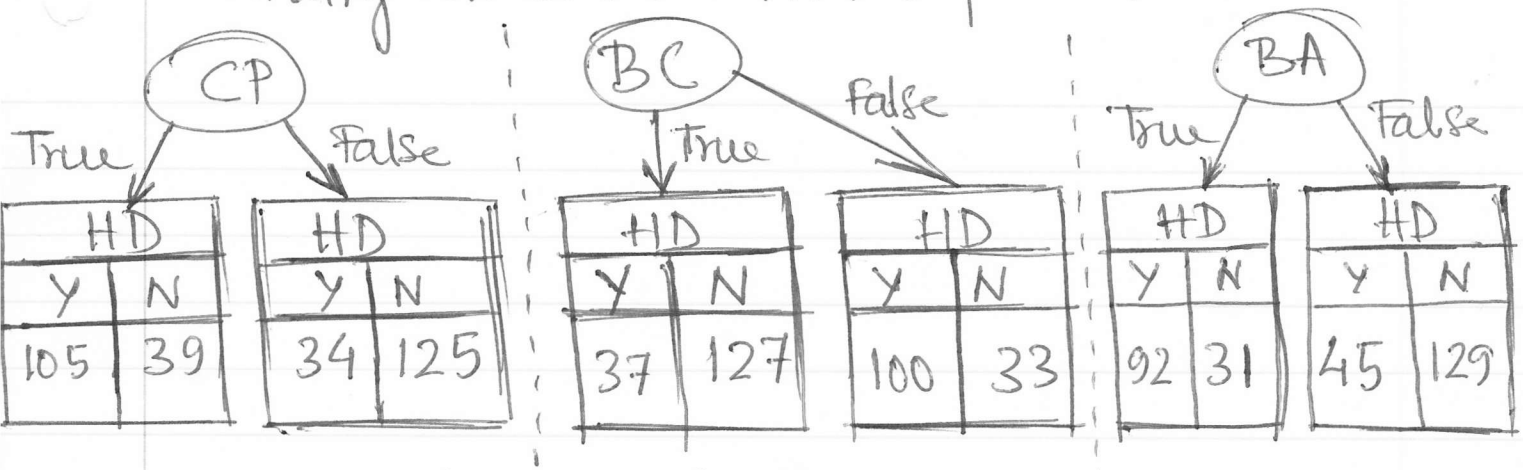
— Data set

Chest pain (CP)	Good Blood Circulation (BC)	Blocked Arteries (BA)	Heart Disease (HD)
NO	NO	NO	NO
YES	YES	YES	YES
YES	YES	NO	NO
YES	NO	???	YES
⋮	⋮	⋮	⋮

- 3 attributes to predict heart disease
- there are 303 patients (rows) in total
- If the % of missing data is very low (< 1%) we can skip rows with missing data.

— Step 1: Finding Root Node

- Finding how each attribute predict HD.



HD	
Y	N
105	39

HD	
Y	N
34	125

HD	
Y	N
37	127

HD	
Y	N
100	33

HD	
Y	N
92	31

HD	
Y	N
45	129

- the nodes (leaf) do not seperate the HD with 100% Certainty — they are impure nodes

— Step 2: measure node impurity — Gini coefficient

- Calculating Gini coefficient for CP:

For 'True' node: $1 - (\text{Probability of Yes})^2 - (\text{Probability of NO})^2$

$= 0.395$ — with 144 patients

For 'false' node $= 0.336$ — with 159 patients

Total Gini coefficient of CP to predict HD

$$= \left(\frac{144}{144+159}\right) 0.395 + \left(\frac{159}{144+159}\right) 0.336$$

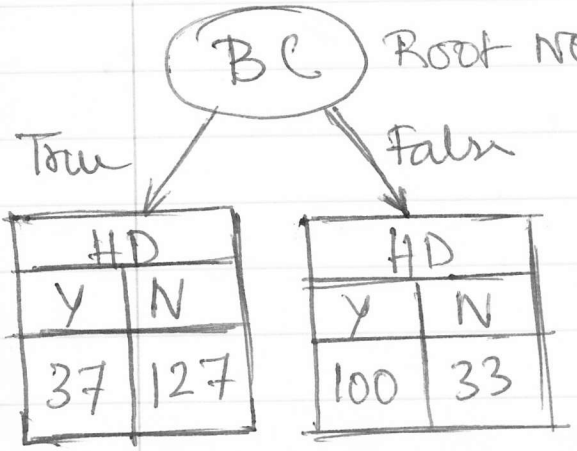
$\Rightarrow$ Gini coefficient for CP $= 0.364$

" " " BC $= 0.360$ and

" " " BA $= 0.381$

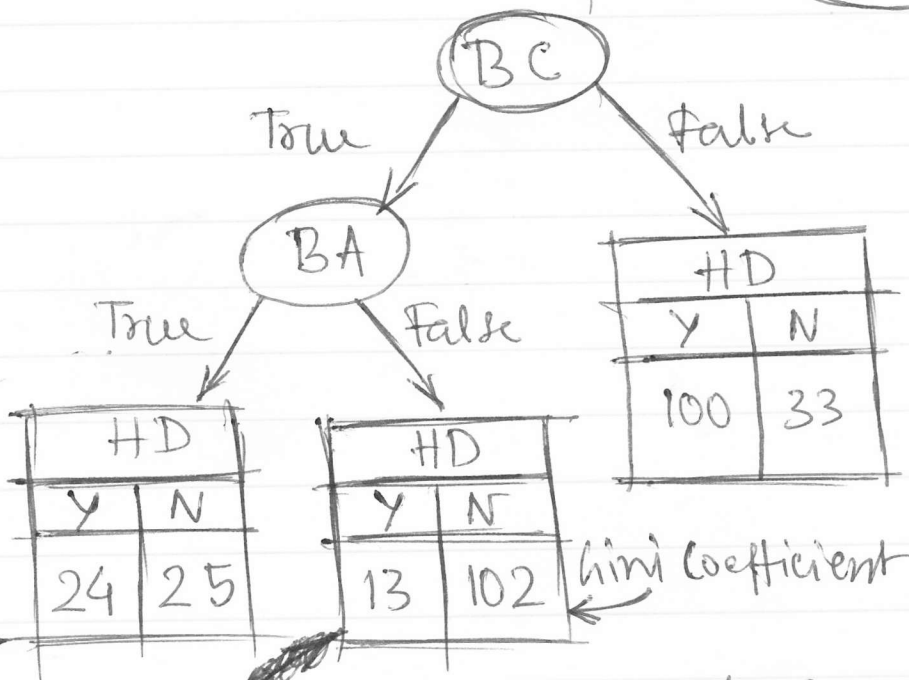
— Step 3: Finding other nodes + leaf nodes

- Pure node has Gini Impurity = 0
- we need to choose the node/attribute with minimum Gini for the Root node — BC

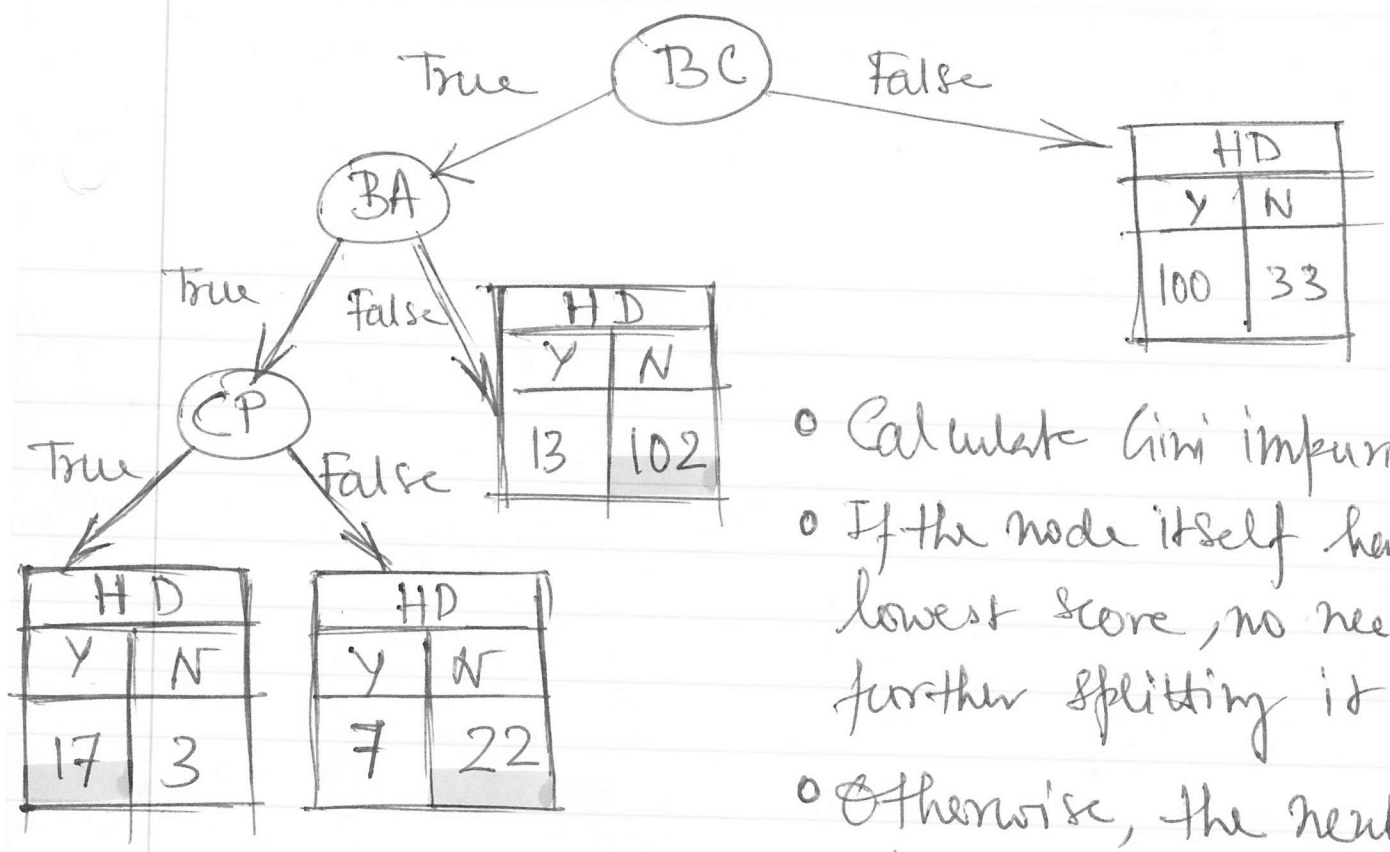


- Find out how well CP and BA seperates these 164 (37+127) patients.

Gini coefficient for CP = 0.3
BA = 0.29

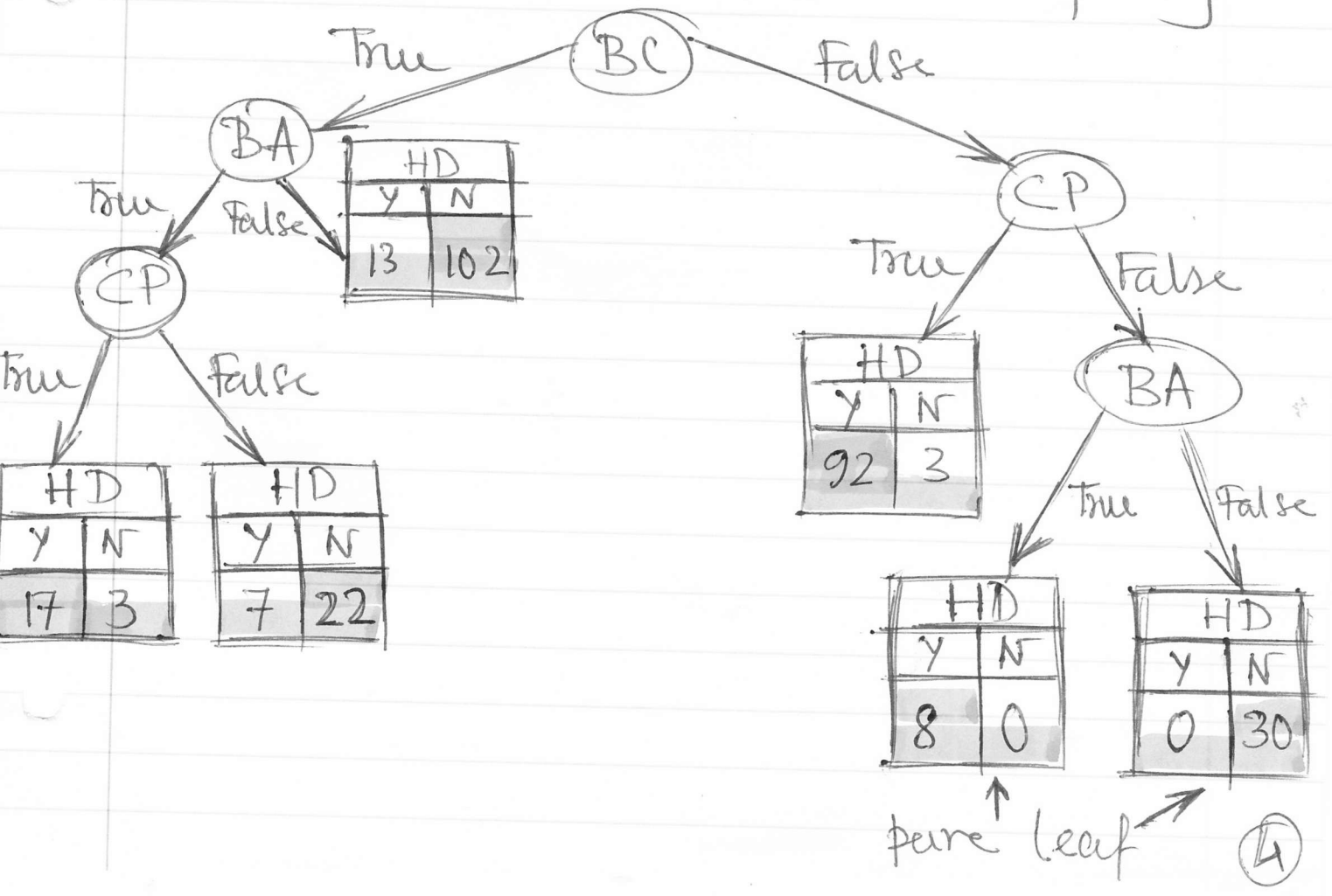


- Apply CP to seperate patients (49) with and without HD



- Calculate Gini impurity
- If the node itself has the lowest score, no need to further splitting it
- Otherwise, the next node will be the one with the lowest Gini impurity value.

Final Tree



Random forest

- Decision trees are, in general, prone to overfitting the data, and as a result do not perform well on a diverse set of data than the training data.
- Step 1: Create a "bootstrapped" dataset.
  - In bootstrapping, we are allowed to pick the same sample (row) from a dataset multiple times.

original Dataset

Chest Pain (CP)	Good Blood Circulation (BC)	Blocked Arteries (BA)	weight	Heart Disease (HD)
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES

Chest Pain	Good Blood Circulation	Blocked Arteries	weight	Heart Disease
YES	YES	YES	180	YES
NO	NO	NO	125	NO
YES	NO	YES	167	YES
YES	NO	YES	167	YES (5)

- Step 2: Create a decision tree using the bootstrap dataset, but using a random subset of column (variable) at each step

---

- let's say we are only using 'Good Blood Circulation' and 'Blocked Arteries' (2 random variables) to create decision tree to predict heart disease.

↓

Say "Good Blood Circulation' is the root node,

- we again randomly select 2 variable: "Chest Pain" and "weight" and so on to build a tree

⇒ Make another bootstrap dataset
↳ build another decision tree from random subset of variables.

⇒ At the end, we will have a variety of decision trees (~100) in the random forest.

Predicting using Random Forest:

- we run the data through each tree and collect their prediction.
- we make the final prediction on the aggrated data (usually) based on maximum probability

- Bagging = Bootstrapping dataset
  + Decision making using aggregated prediction from individual tree.

Evaluating a Random Forest

- Usually, about $\frac{1}{3}$ of the original data do not end up in the bootstraped dataset.
  — called Out-of-Bag Dataset
- Out-of-Bag dataset is used to
  - evaluate the performance of the Random Forest
  - optimize the number of variables used from the bootstraped dataset to build decision trees
- Incorrectly classified Out-of-Bag Samples are called Out-of-Bag Error.
