Machine Learning 2 - Decision Trees

Ramesh Kestur, IIITB

Rameshkestur@iiitb.ac.in

Public profile:
Linkedin
Google Scholar
IIITB

4-Nov-23

Topics

- Linear regression and Decision trees - Comparison
- Building a decision tree
  - Deciding the feature to split
    - Gini index and Entropy
- Regression decision trees
  - Deciding the feature to split: Mean square Error
- Over fitting
  - Bagging and boosting
- Random forest
  - Boot strapping and aggregation (Bagging)
  - Boot strapping
    - OOB error / Random sampling with replacement
  - Tree correlation
  - Split variable randomization
- Code demo/walkthrough

Linear regression and Decision trees

Linear Regression assumptions

- Assumes Linear relation between explanatory variables and predicted variables
- The features are independent
- There should be no multicollinearity between independent variables

Decision trees

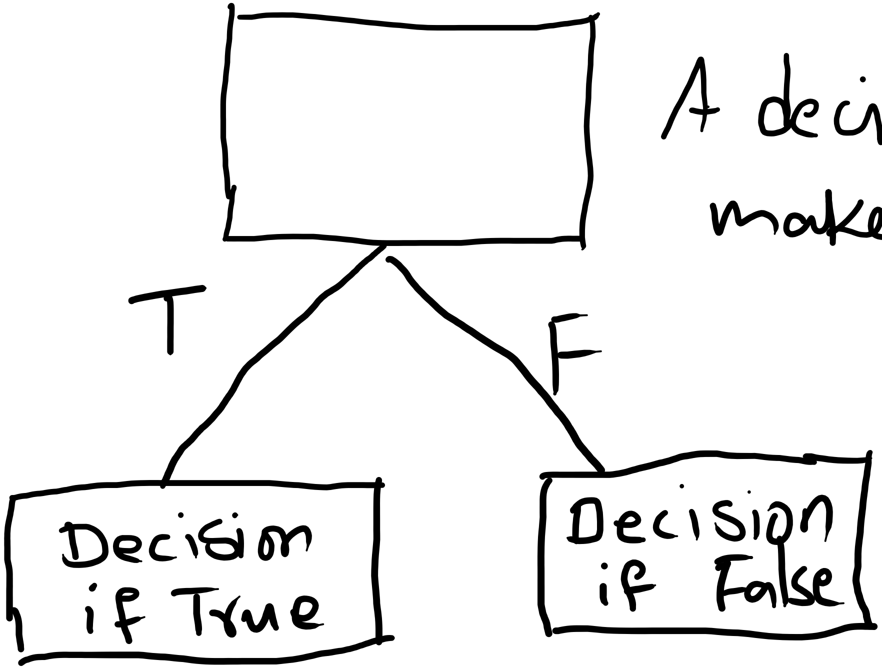
- Supports non-linearity
- The features need not essentially be normalized
- Provides interpretability
- Categorical variables

Decision Trees

-

Decision tree is a popular tool for classification and prediction. A Decision tree is a flowchart like tree 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.

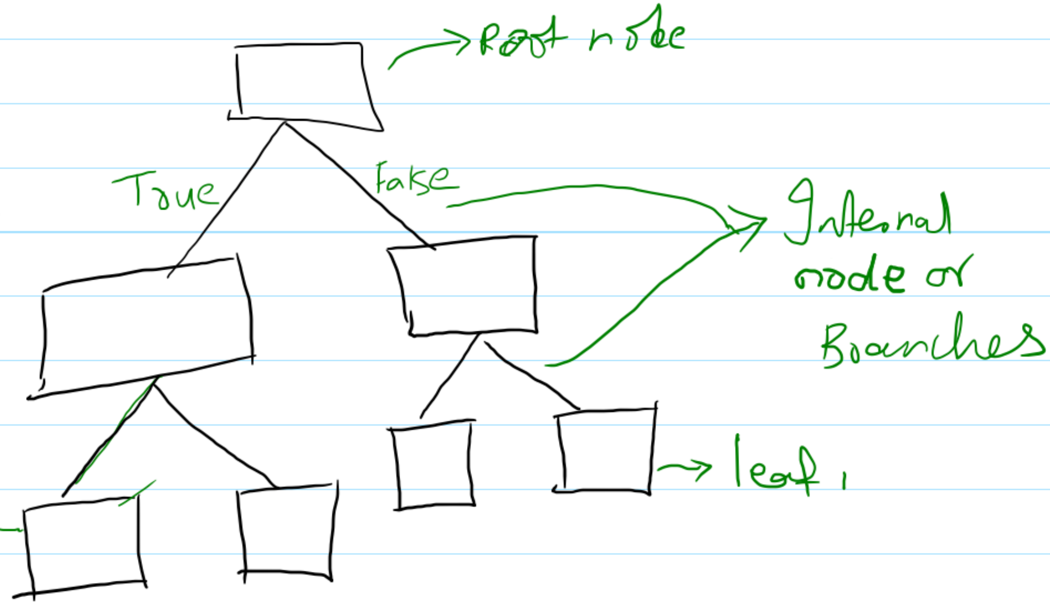
Decision tree is a popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



A decision tree makes a statement

Terminologies

Convention
Statement = True
go to left.
Statement = False
go to right.



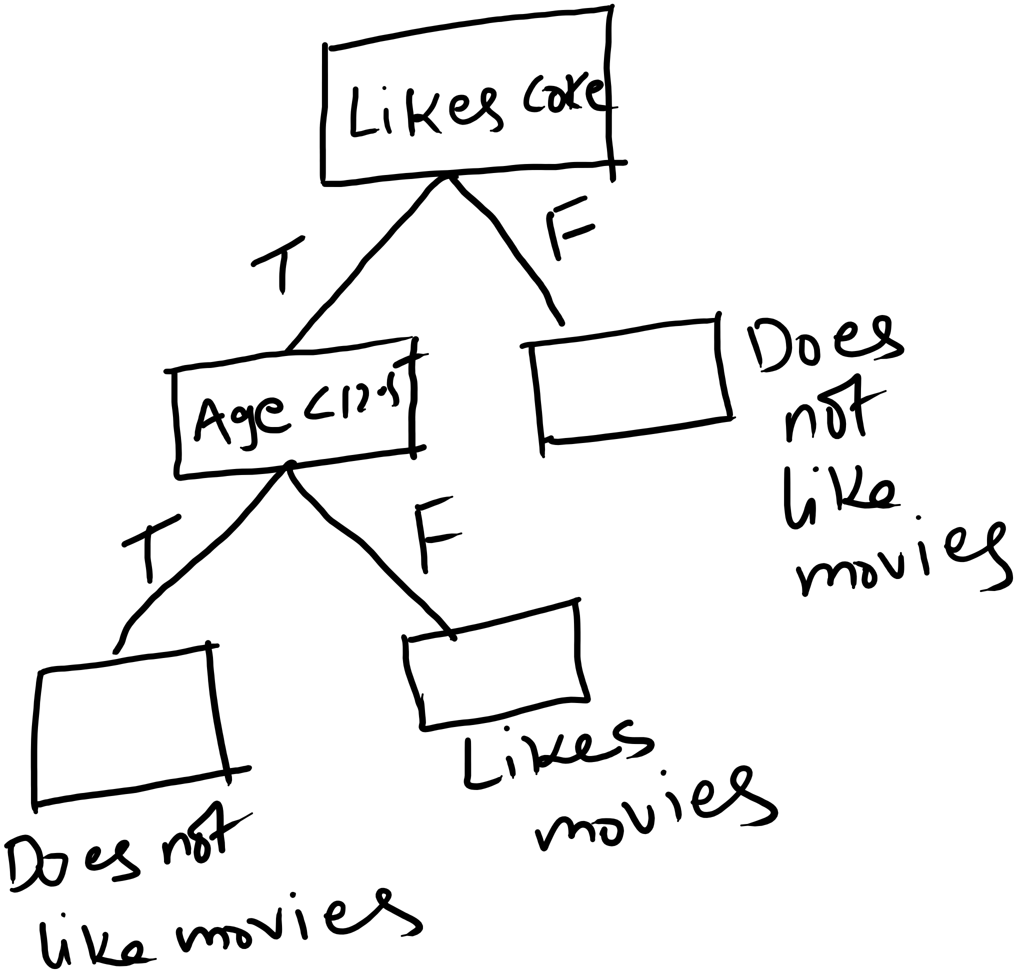
Leaves have arrows pointing to them but no arrows away from them

Branches have arrows pointing to them & arrows away from them

Decision trees...

- When a decision tree classifies things into categories it is called a classification tree.
- When it predicts a numerical value it is called a regression tree
- Datatypes can be mixed in the decision tree
- Numeric thresholds can be same for the same data.
- Data need not be necessarily normalized
- The final classifications can be repeated

Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Adapted from : https://www.youtube.com/watch?v=_L39rN6gz7Y

Which one of these features to choose as root node?

Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Gini Index or Information Gain

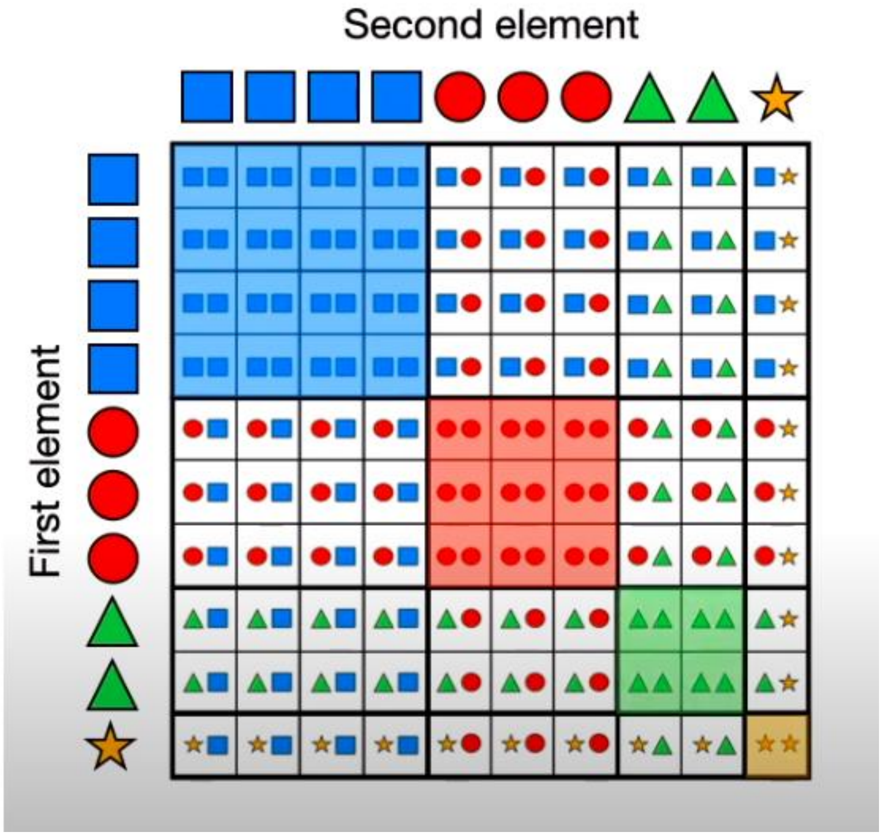
Where to split ?

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

Measure of ‘impurity’?

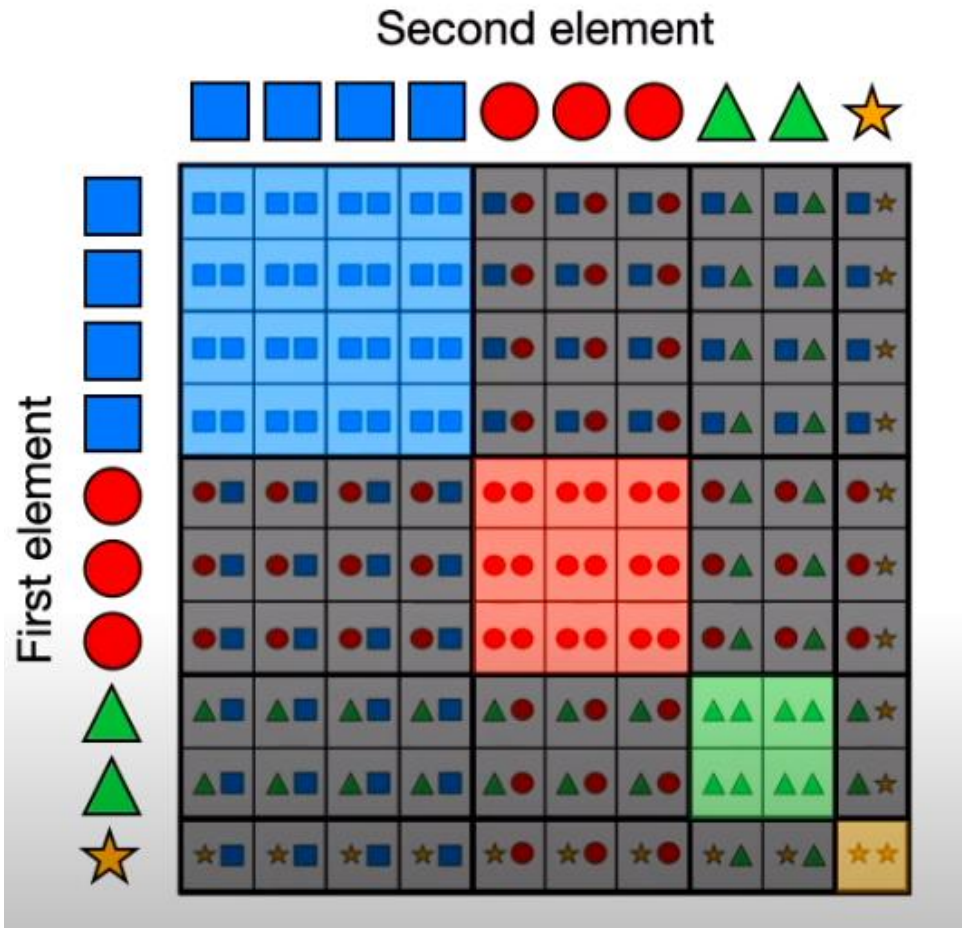
- If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1,
  where, 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes A Gini Index of 0.5 denotes equally distributed elements into some classes.

Where to split ?

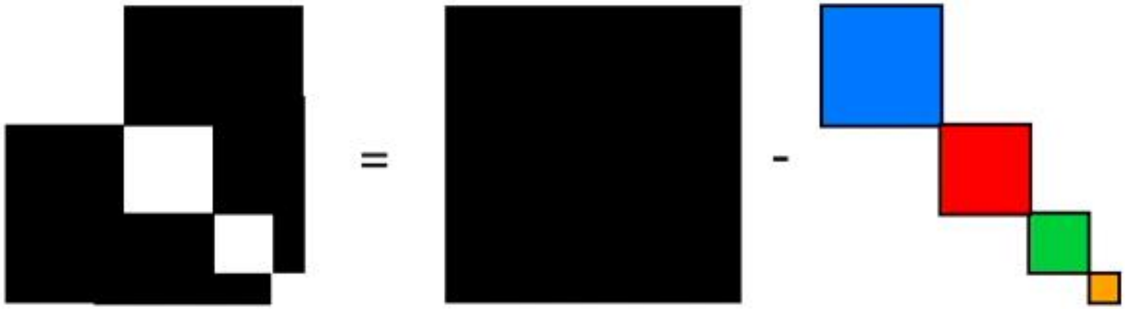


adapted from : https://www.youtube.com/@SerranoAcademy/videos

Where to split ?



Where to split ?



P(Both different) = P(Anything) - P(Both equal)

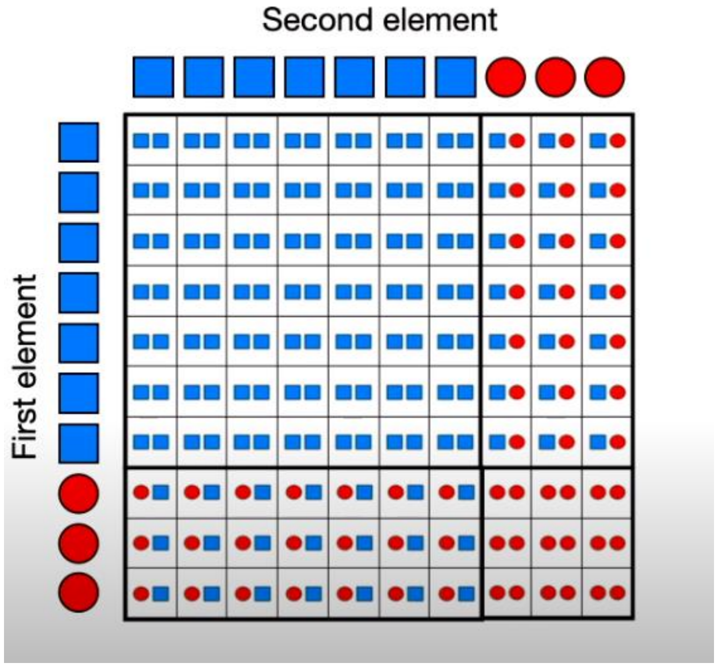
= 1 - P(Both blue) - P(Both red) - P(Both green) - P(Both yellow)

$$= 1 - \frac{16}{100} - \frac{9}{100} - \frac{4}{100} - \frac{2}{100}$$

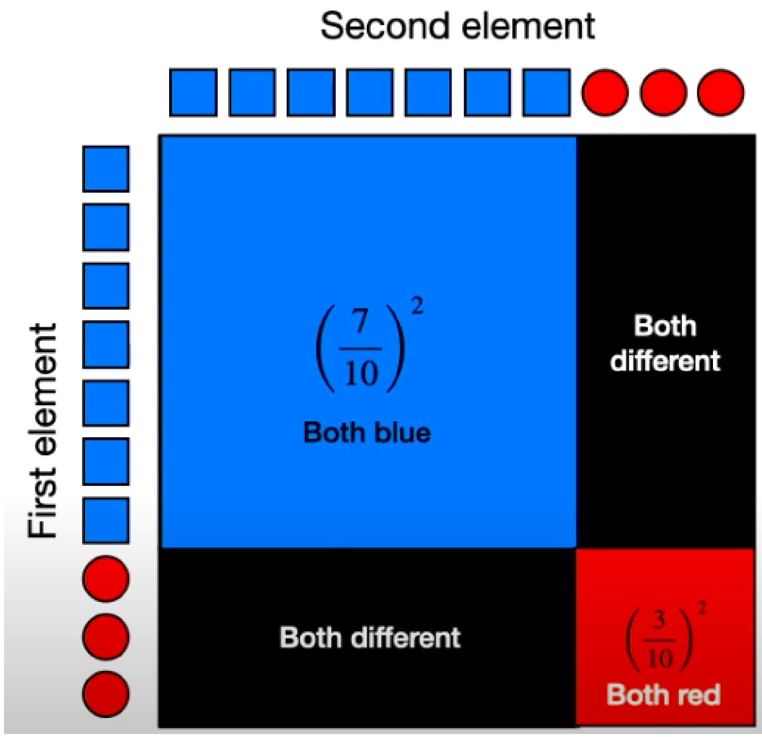
$$= 0.7$$

Where to split ?

Consider another dataset



Where to split ?



$$\text{P(Both different)} = \text{P(Anything)} - \text{P(Both equal)}$$

$$= 1 - \text{P(Both blue)} - \text{P(Both red)}$$

$$= 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2$$

$$= 1 - 0.7^2 - 0.3^2$$

$$= 1 - 0.49 - 0.09$$

$$= 0.42$$

Where to split ?

Generalizing, we get

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

where n = no of classes

P_i = Probability of the class

Proof of Gini index

$$\sum_{i=1}^{c} p_i (1 - p_i)$$

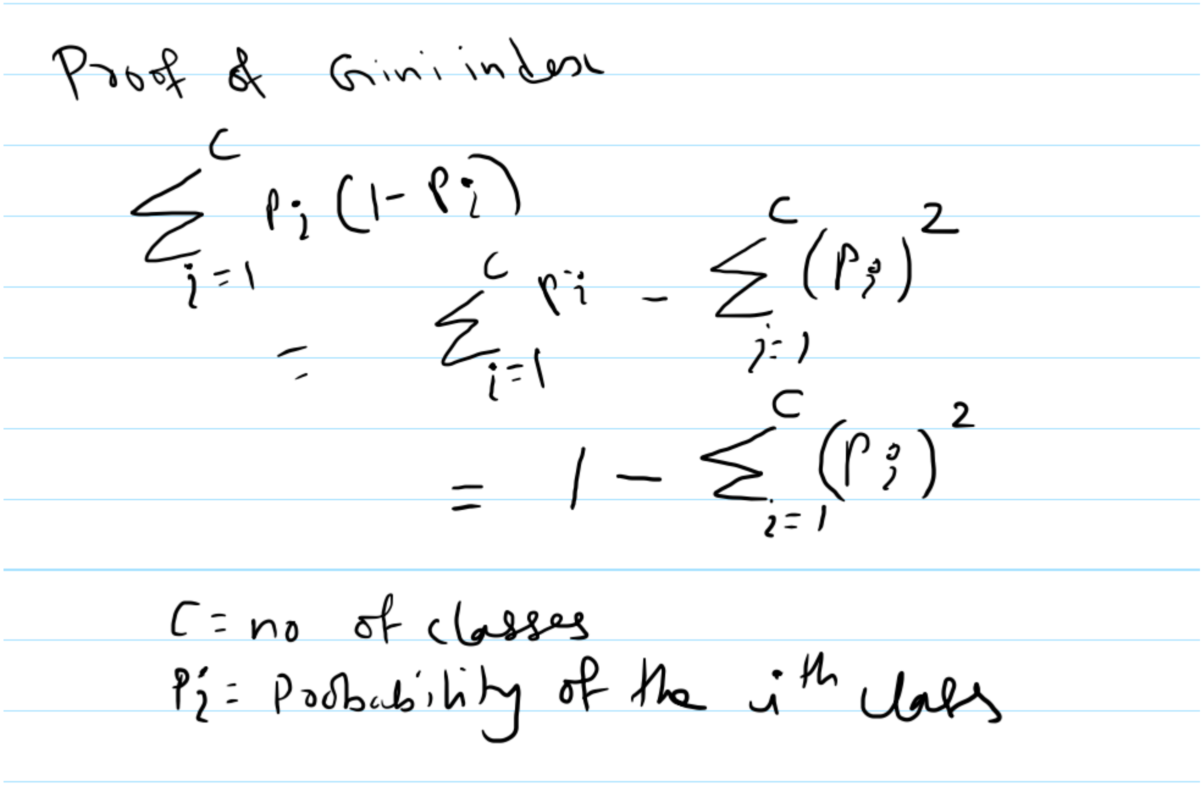
$$= \sum_{i=1}^{c} p_i - \sum_{i=1}^{c} (p_i)^2$$

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

c = no of classes

p_i = Probability of the ith class

Where to split ?



Proof of Gini index

$$\sum_{i=1}^{c} p_i(1-p_i)$$

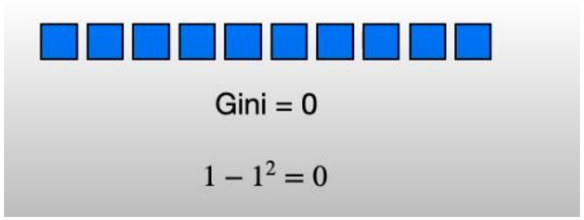
$$= \sum_{i=1}^{c} p_i - \sum_{i=1}^{c}(p_i)^2$$

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

c = no of classes

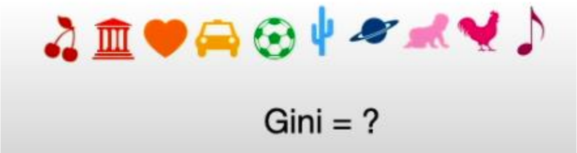
p_i = Probability of the i^{th} class

Where to split ?



Gini = 0

$1 - 1^2 = 0$



Gini = ?

$$= 1 - (.1)^2 - (.1)^2 - (.1)^2 \cdots\cdots (.1)^2$$

$$= 1 - (.01)10 = 1 - 0.1$$

$$= 0.9$$

Entropy

Entropy: It is used to measure the impurity or randomness of a dataset.

Logarithm of fractions gives a negative value and hence a '-' sign is used in entropy formula to negate these negative values.

Entropy is not preferred due to the 'log' function as it increases the computational complexity.

$$\text{Entropy} = -\sum_{i=1}^{n} p_i * \text{Log}_2(p_i)$$

Information gain : A measure of how much information a feature provides about a class. Information gain helps to determine the order of attributes in the nodes of a decision tree.

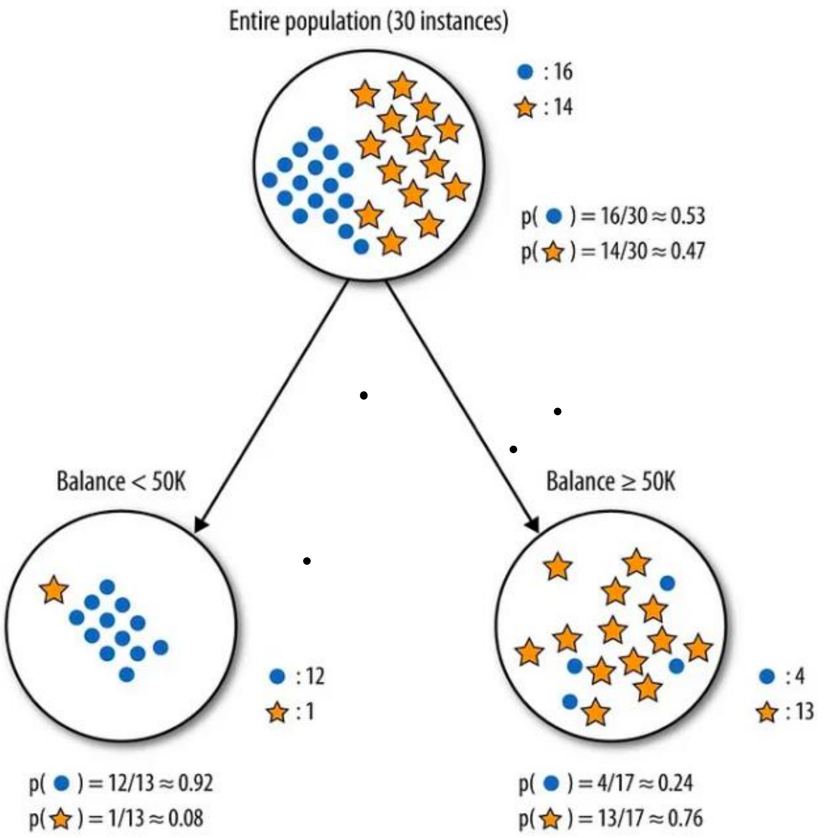
$$\text{Gain} = E_{\text{parent}} - E_{\text{children}}$$

E_{parent} is the entropy of the parent node and E_{children} is the average entropy of the child nodes

Consider an example where of building a decision tree to predict whether a loan given to a person would result in a write-off or not. The entire population consists of 30 instances. 16 belong to the write-off class and the other 14 belong to the non-write-off class. We have two features, namely “Balance” that can take on two values -> “< 50K” or “>50K” and “Residence” that can take on three values -> “OWN”, “RENT” or “OTHER”

Source: https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8#:~:text=So%20if%20we%20had%20a,example%20using%20the%20formula%20above.

Balance



Source: Provost, Foster; Fawcett, Tom. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

$$E(Parent) = -\frac{16}{30}\log_2\left(\frac{16}{30}\right) - \frac{14}{30}\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Balance < 50K) = -\frac{12}{13}\log_2\left(\frac{12}{13}\right) - \frac{1}{13}\log_2\left(\frac{1}{13}\right) \approx 0.39$$

$$E(Balance > 50K) = -\frac{4}{17}\log_2\left(\frac{4}{17}\right) - \frac{13}{17}\log_2\left(\frac{13}{17}\right) \approx 0.79$$

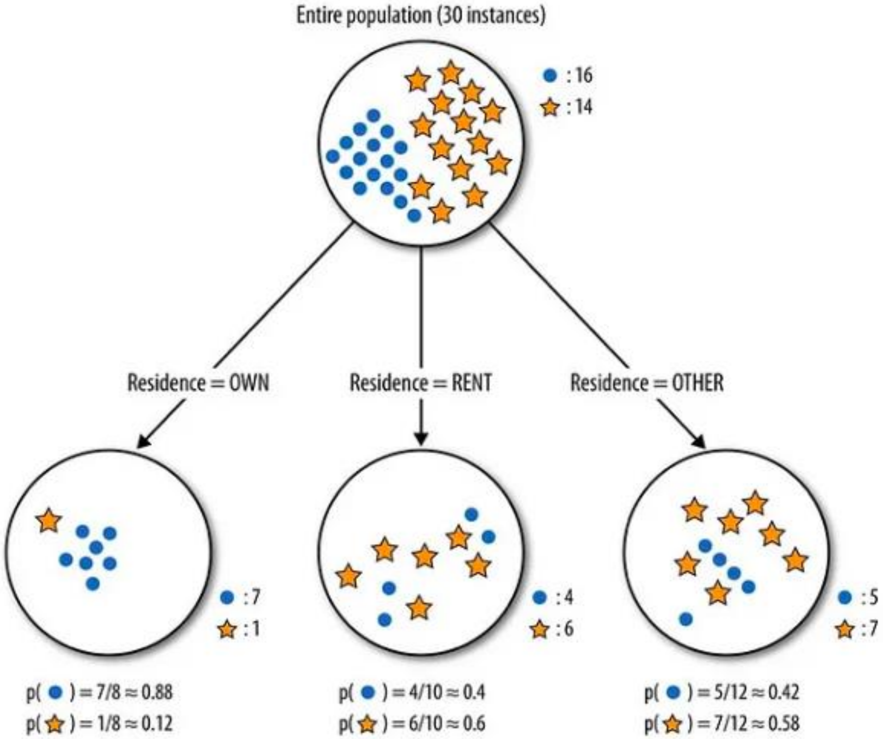
Weighted Average of entropy for each node:

$$\begin{aligned} E(Balance) &= \frac{13}{30} \times 0.39 + \frac{17}{30} \times 0.79 \\ &= 0.62 \end{aligned}$$

Information Gain:

$$\begin{aligned} IG(Parent, Balance) &= E(Parent) - E(Balance) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$

Residence


Entire population (30 instances)
● : 16
★ : 14
Residence = OWN
Residence = RENT
Residence = OTHER
● : 7
★ : 1
● : 4
★ : 6
● : 5
★ : 7
p(●) = 7/8 ≈ 0.88
p(★) = 1/8 ≈ 0.12
p(●) = 4/10 ≈ 0.4
p(★) = 6/10 ≈ 0.6
p(●) = 5/12 ≈ 0.42
p(★) = 7/12 ≈ 0.58

$$E(Residence = OWN) = -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

$$E(Residence = RENT) = -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \approx 0.97$$

$$E(Residence = OTHER) = -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \approx 0.98$$

Weighted Average of entropies f or each node:

$$E(Residence) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

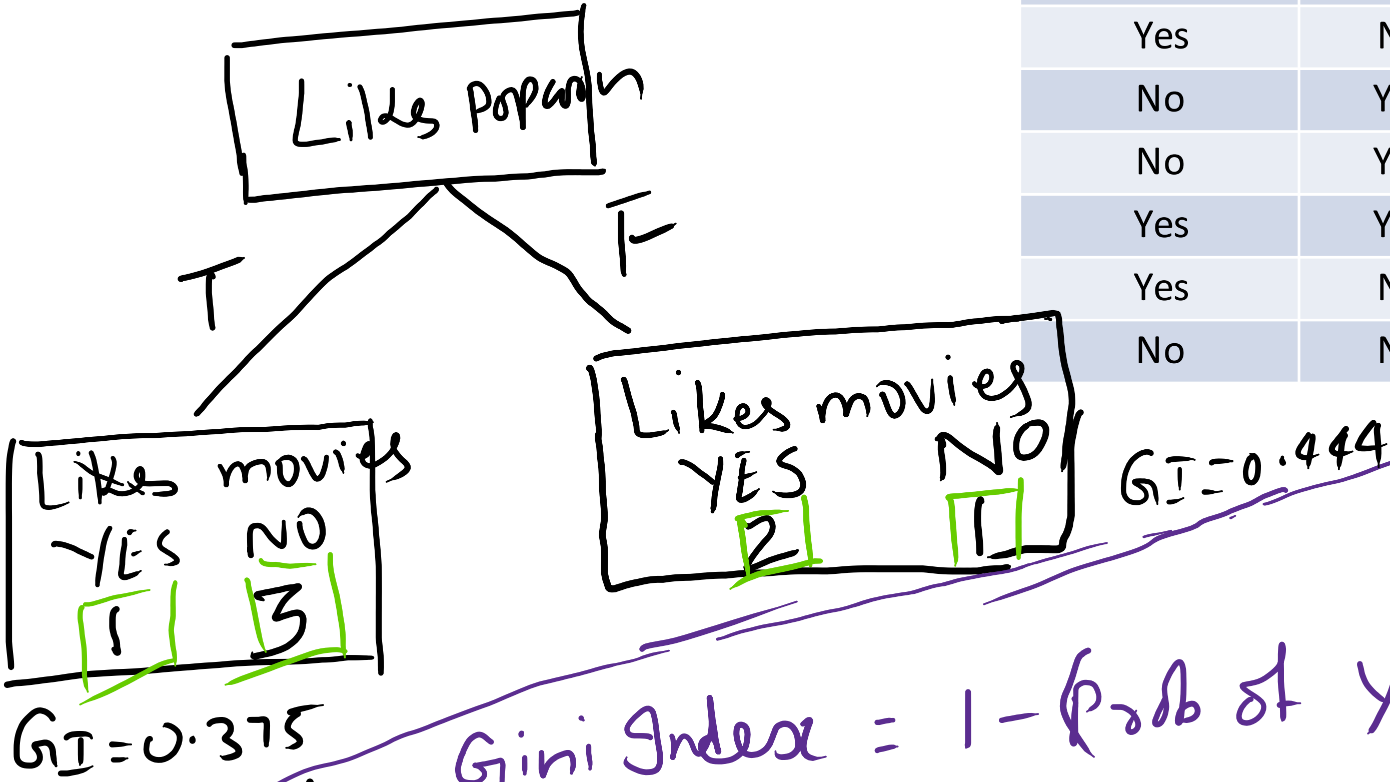
Inf ormation Gain:

$$\begin{aligned} IG(Parent, Residence) &= E(Parent) - E(Residence) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

The feature Balance has a > IG than Residence
hence Balance ✓

Entropy is not preferred due to the ‘log’ function as it increases the computational complexity. Let's get back to building our decision tree using GINI impurity

Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



G

$$\text{Gini Index} = 1 - (\text{Prob of yes})^2 - (\text{Prob of No})^2$$

$$= 1 - \left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2$$

$$= 0.375$$

Similarly for leaf 2

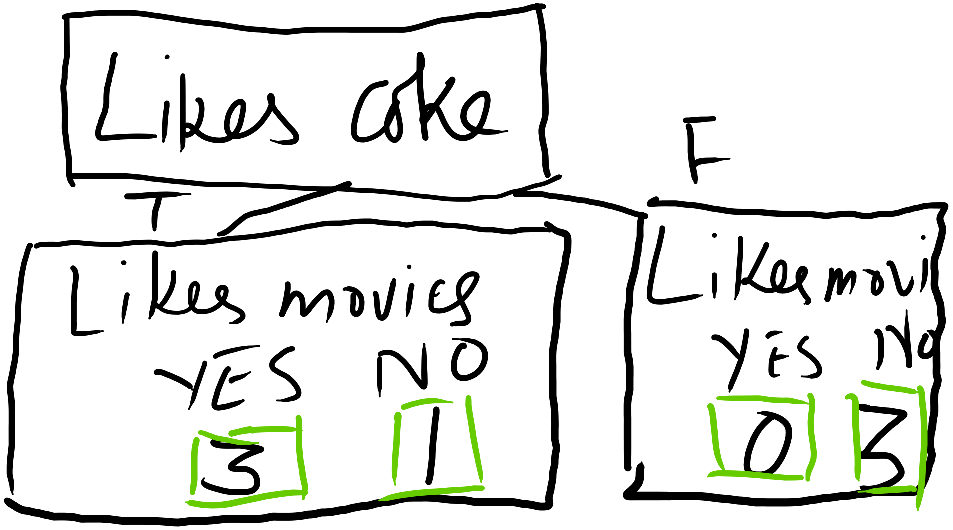
$$GI = 1 - \left(\frac{2}{3}\right)^2 - \left(1/3\right)^2$$

$$= 0.444$$

Total Gini impurity = weighted average of GI for the leaves

$$= \frac{4}{7} \times 0.375 + \frac{3}{7} \times 0.444$$

$$= 0.405$$



Similarly GI for likes coke is

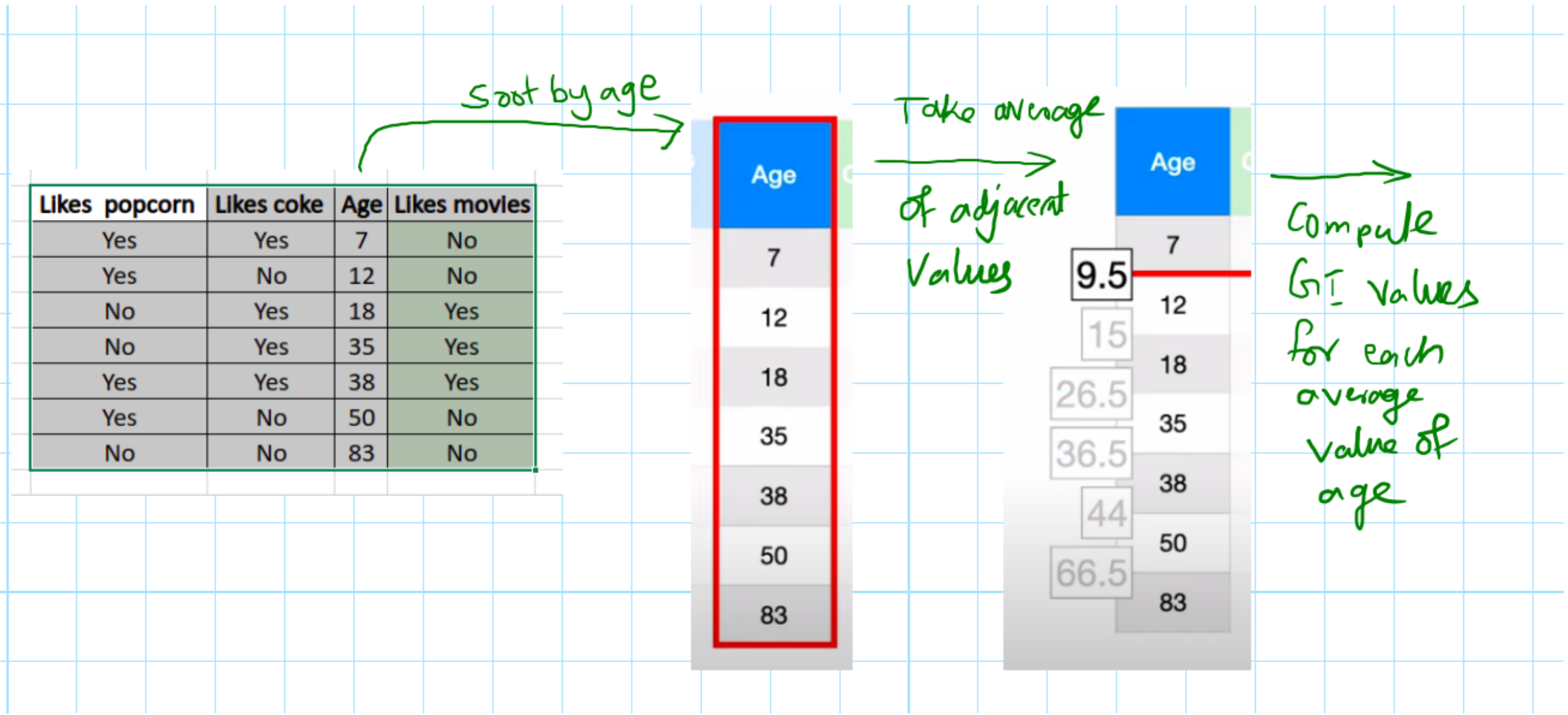
First leaf, $GI = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = .375$

Second leaf, $GI = 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0$

Total GI for likes coke $= \left(\frac{4}{7}\right) \cdot .375 - 0$

$= 0.214$ //

To compute GI for age

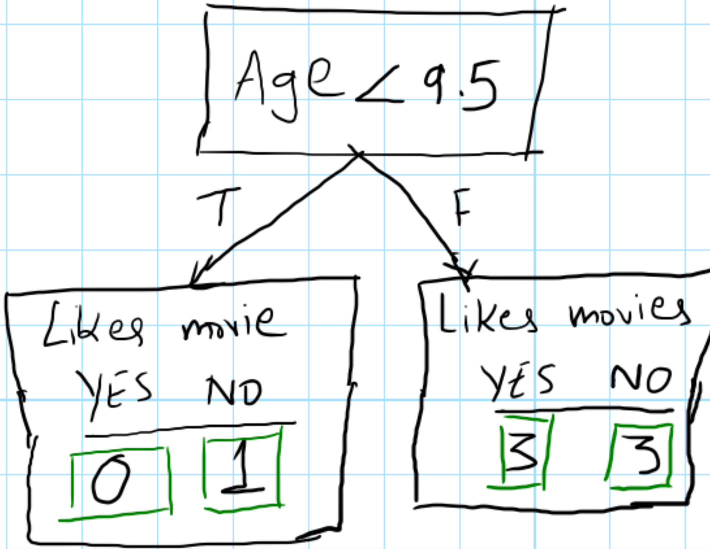


Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Sort by age

Take average of adjacent Values

Compute GI values for each average value of age



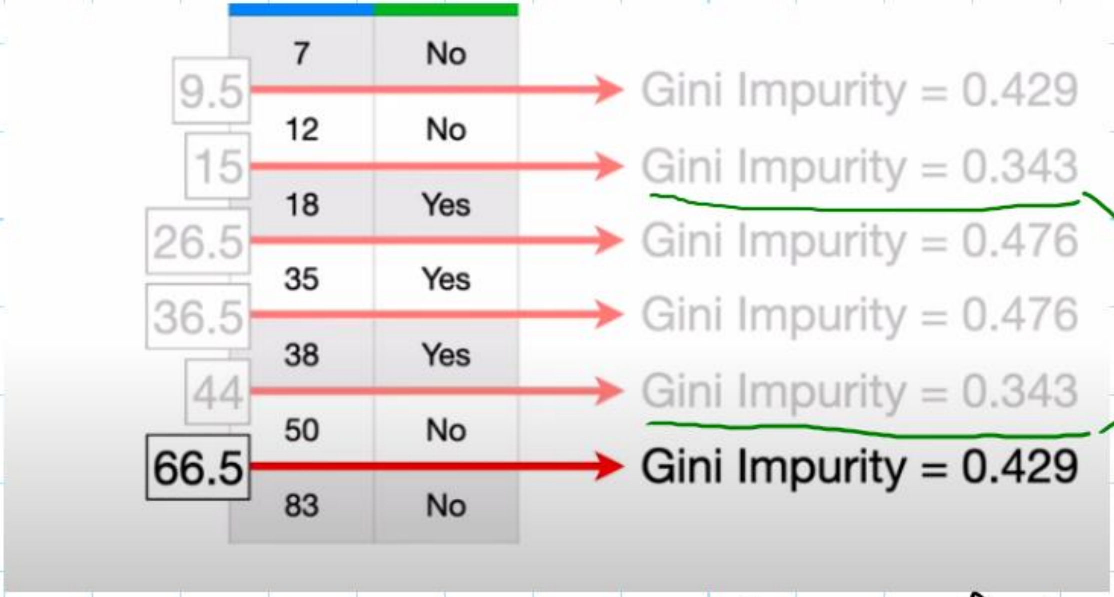
$\rightarrow \text{Impurity} = 1-\left(\frac{3}{3+3}\right)^2-\left(\frac{3}{3+3}\right)^2$

$= 0.5$

GI = 0

$\text{Impurity} = 1-\left(\frac{0}{1}\right)^2-\left(\frac{1}{1}\right)^2 = 0$

$\text{Total impurity} = \frac{1}{(1+6)}\times 0 + \frac{6}{(1+6)}\times 0.5 = \underline{\underline{0.429}}$



7	No
12	No
18	Yes
35	Yes
38	Yes
50	No
83	No

9.5 → Gini Impurity = 0.429

15 → Gini Impurity = 0.343

26.5 → Gini Impurity = 0.476

36.5 → Gini Impurity = 0.476

44 → Gini Impurity = 0.343

66.5 → Gini Impurity = 0.429

these two values have the lowest GI either of them can be chosen

Similarly compute the GI for all candidate values

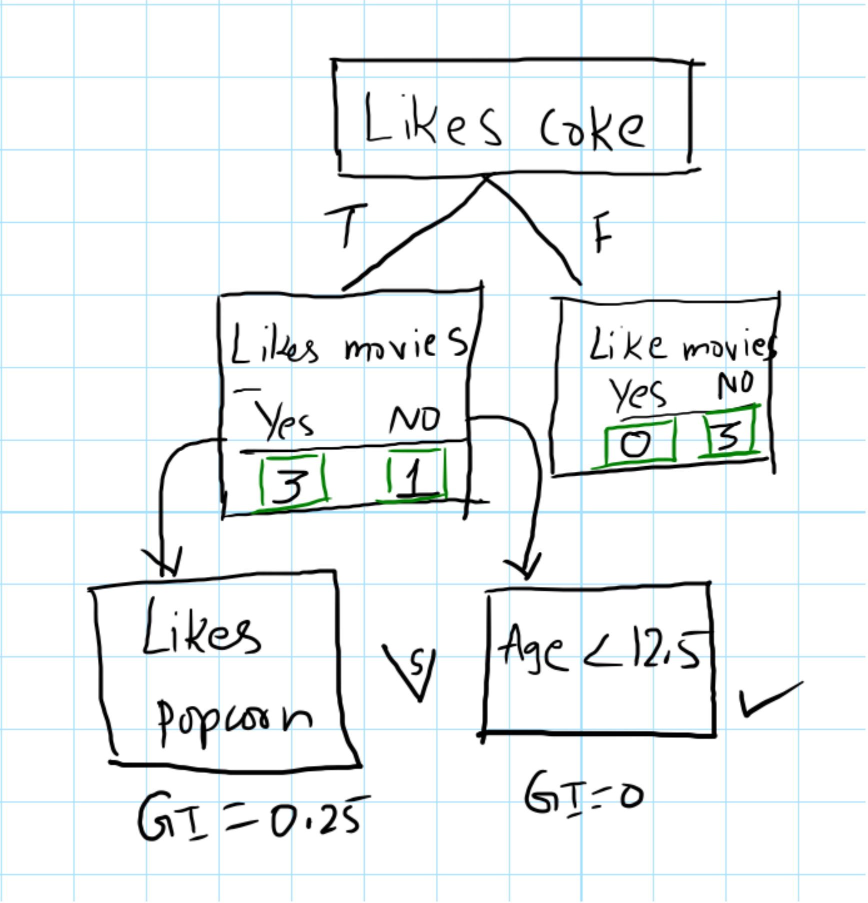
The GI of Like popcorn = 0.405

$\boxed{\text{Like Coke} = 0.214}$

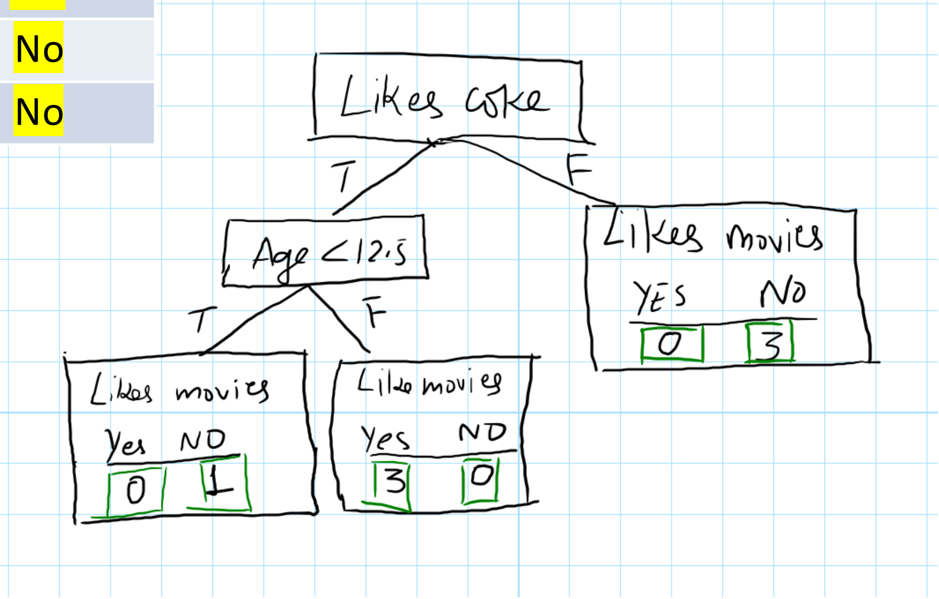
Age < 15 = 0.343

Hence choose Like coke as the root node

Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

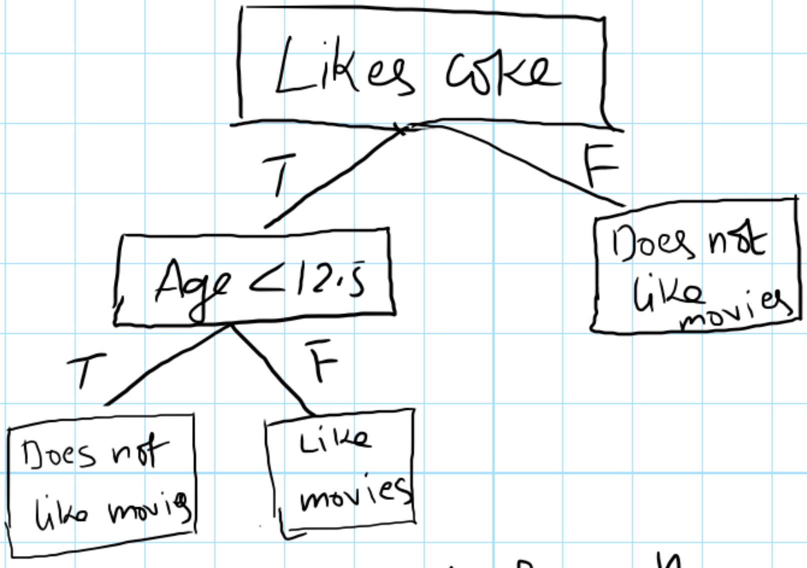


Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



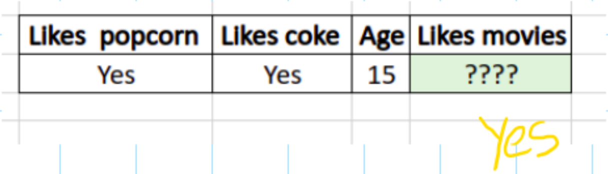
Need to assign output values for each leaf

Final tree!!!



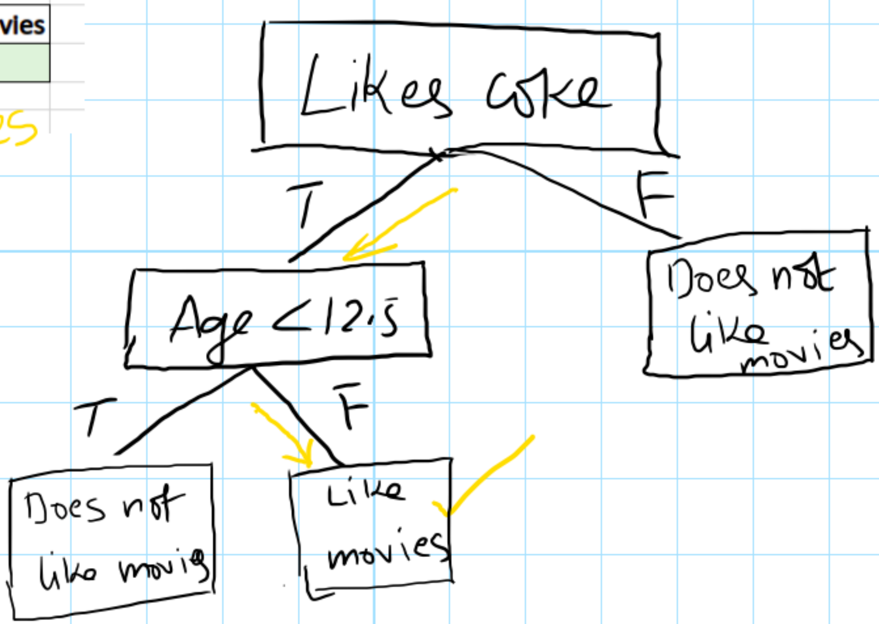
The output of a leaf is the Category with most votes

Prediction.



Likes popcorn	Likes coke	Age	Likes movies
Yes	Yes	15	????

Yes



As a problem usually has a large set of features, it results in large number of split, which in turn gives a huge tree. Such trees are *complex and can lead to overfitting.*

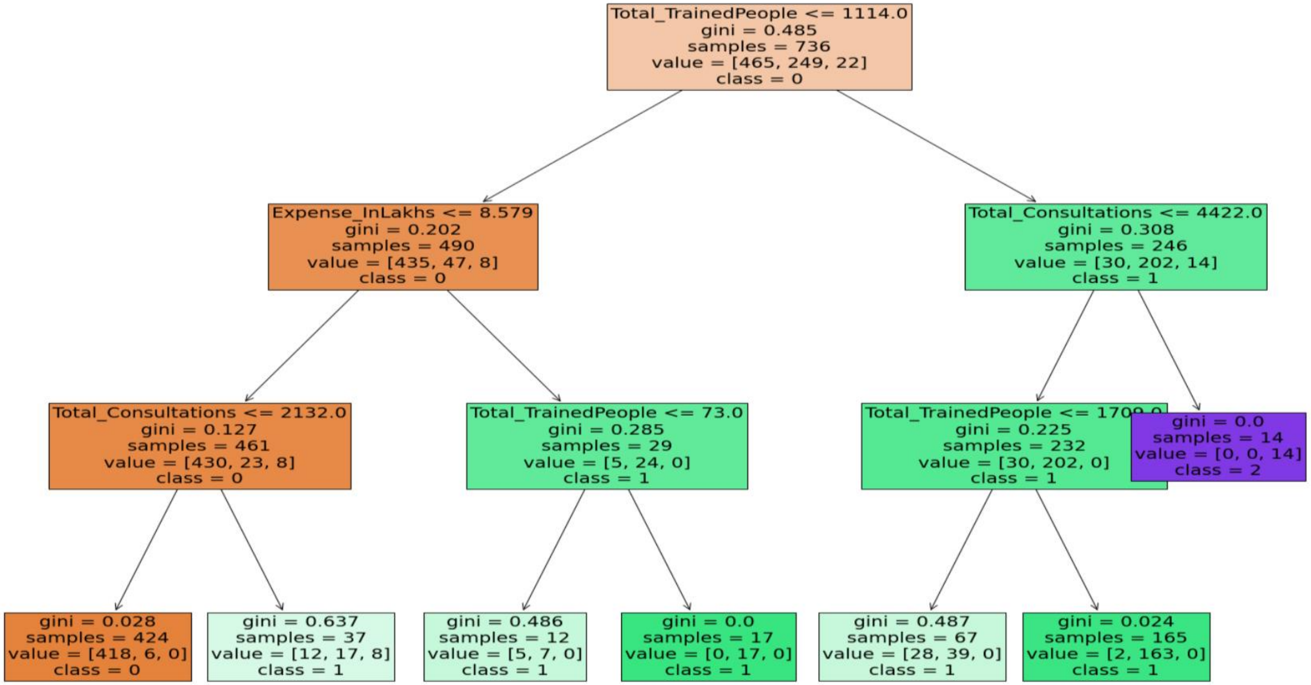
Pruning

Remove **the branches that make use of features having low importance.** This way,we reduce the complexity of tree, and thus increasing its predictive power by reducing overfitting.

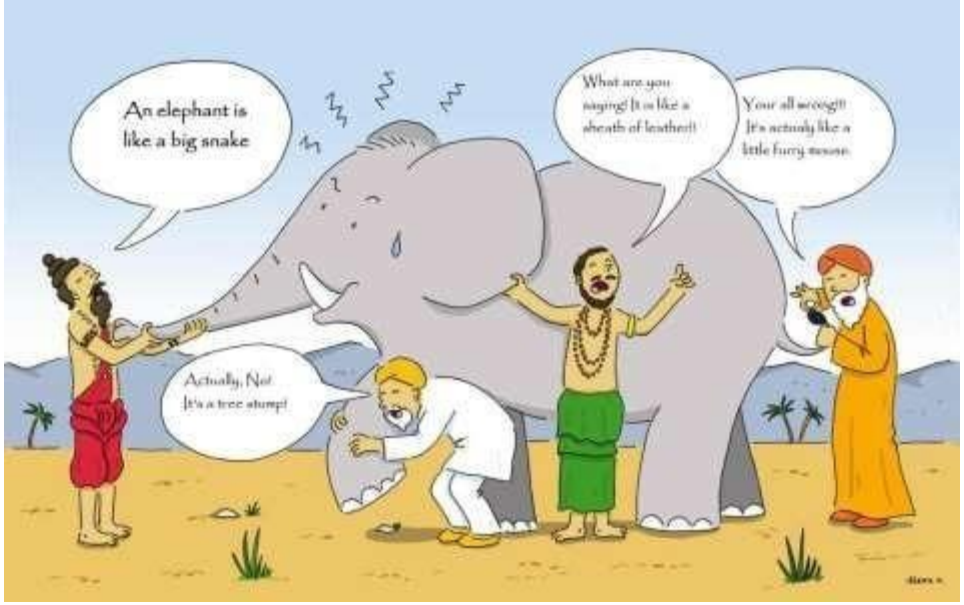
Truncation

Stop the tree while it is still growing so that it may not end up with leaves containing vary less data-points, one way to do this is by setting the minimum number

Visualizing a decision tree



Ensemble learning is a machine learning paradigm where multiple models (often called “weak learners”) are trained to solve the same problem and combined to get better results.


An elephant is like a big snake
Actually, No! It's a tree stump!
What are you saying! It is like a sheath of leather!!
Your all wrong!! It's actually like a little furry mouse.

https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Ensemble Learning

- A single model, also known as a base or weak learner, may not perform well individually due to high variance or high bias
- Wisdom of crowds
    - the decision-making of a larger group of people is typically better than that of an individual expert
- When weak learners are aggregated, they can form a strong learner
- The combination reduces bias or variance, yielding better model performance

- Decision Trees overfitting (High variance low bias) when the trees are not pruned
- underfitting (low variance and high bias) when the trees are small

Ensemble Methods

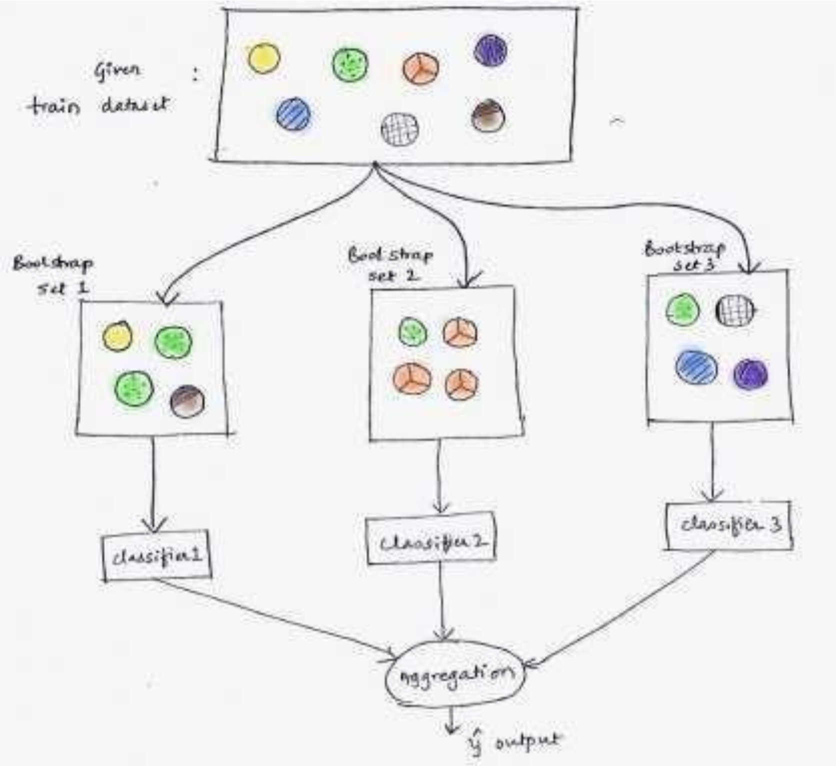
Bagging

- Weak learners are trained in parallel
- Typically used on weak learners that exhibit high variance and low bias
- The combination reduces bias or variance, yielding better model performance

Boosting

- Weak learners are trained in series
- Leveraged when low variance and high bias is observed.

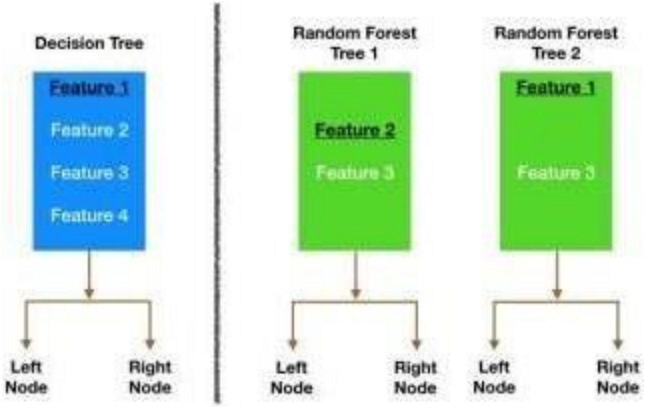
Bagging, that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process



https://learn.upgrad.com/v/course/515/session/77070/segment/431269

Bagging (Bootstrap Aggregation) — Decision trees are very sensitive to the data they are trained on — small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees.

Feature Randomness — In a normal decision tree, while splitting a node, we consider all the features and pick the one that produces the most separation. In contrast, each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification.

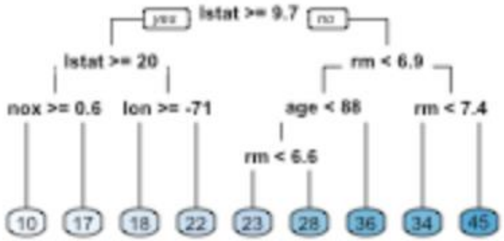


- Out of bag (OOB) error
- Tree correlation
- Split variable randomization

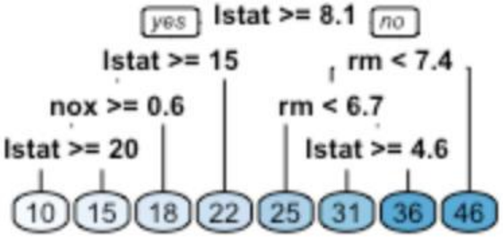
Tree correlation

- Split variable randomization

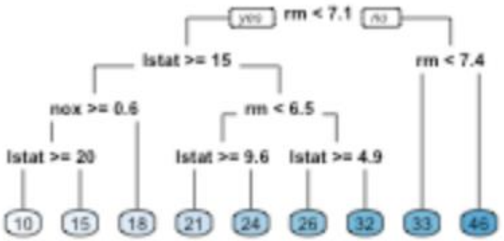
Decision Tree 1



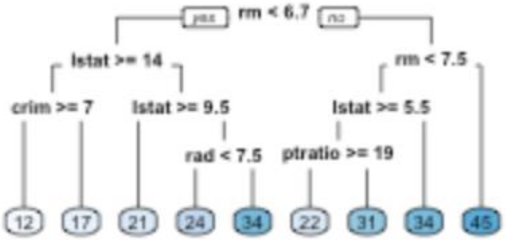
Decision Tree 2



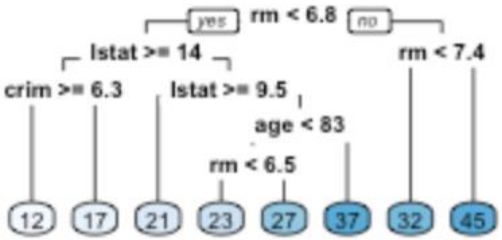
Decision Tree 3



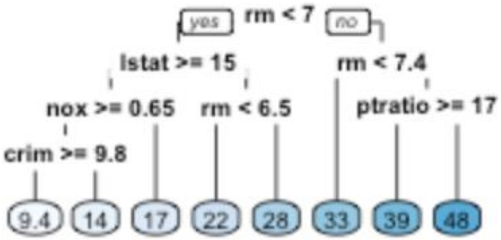
Decision Tree 4



Decision Tree 5



Decision Tree 6



Tree correlation

- Split variable randomization

Six decision trees with different bootstrapped samples of the Boston housing data are created The top of the trees all have a very similar structure, Although there are 15 predictor variables to split on, all six trees have both *lstat* and *rm* variables driving the first few splits.

No of random variables to chose

$m = P/3$ for regression

$m = \sqrt{p}$ for classification

m is a tunable parameter

Thank You