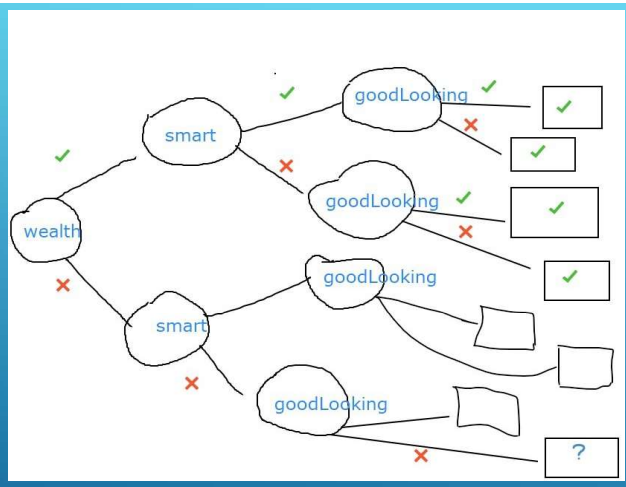
DECISION TREE CLASSIFICATION

Associate Professor Yachai Limpiyakorn, Ph.D.

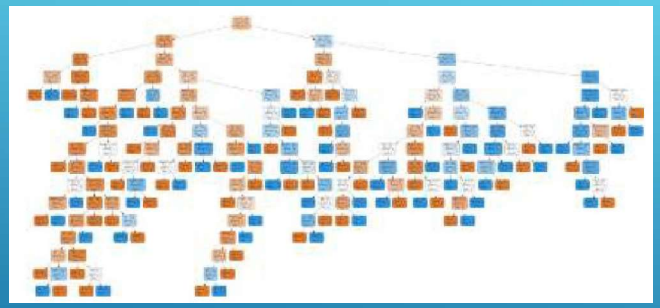


GENERAL DECISION MAKING TREE

PART 1: DECISION TREE CLASSIFIERS

- A tree-like diagram illustrates all possible decision alternatives and the corresponding outcomes.
- Starting from the root of a tree,
  - **internal** node represents the basis on which a decision is made;
  - each **branch** of a node represents how a choice may lead to the next nodes;
  - terminal node, **leaf**, represents the outcome produced.
  - Paths from root to leaves represent **classification rules**

1. Tree Construction



2. Tree Pruning

DECISION TREE LEARNING

Rain	Windy	Xtreme windy	Decision
0	0	0	Nothing
0	0	1	Nothing
0	1	0	Nothing
0	1	1	Nothing
1	0	0	Umbrella
1	0	1	Stay home
1	1	0	Rain jacket
1	1	1	Stay home

Occam's Razor

สมมติฐานที่สั้นกว่าที่สามารถอธิบายข้อมูลได้เหมือนกัน
จะเป็นสมมติฐานที่ดีกว่า

the simplest explanation is usually the best one

~raining → don't bring anything
raining and extremely windy → stay home
raining and ~windy → umbrella
raining and windy → rain jacket

TREE CONSTRUCTION

https://medium.com/@ml.at.berkeley/machine-learning-crash-course-part-5-decision-trees-and-ensemble-models-dcc5a36af8cd

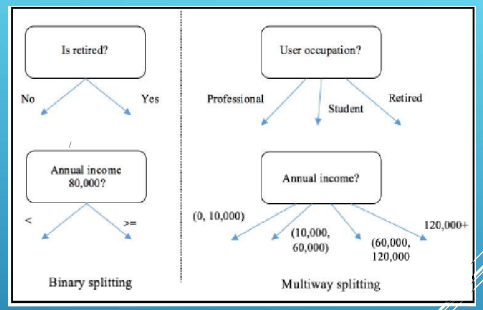
Tree Construction Alternatives

Select the best attribute using Attribute Selection Measures(ASM) to split the records.

Make that attribute a decision node and breaks the dataset into smaller subsets.

Starts tree building by repeating this process recursively for each child until one of the condition will match:

- All the tuples belong to the same class.
- There are no more remaining attributes.
- There are no more instances.



BASIC IDEA OF DECISION TREE ALGORITHM (TOP-DOWN CONSTRUCTION)

Multi-way Split

- Information Gain – ID3 [Ross Quinlan]
- Gain ratio – C4.5 [Ross Quinlan]

Binary Split

- GINI – CART (Classification and Regression Tree)

SPLIT MEASURE/ ASM

ENTROPY/ INFORMATION

กำหนด message M ประกอบด้วยค่าที่เป็นไปได้ $\{m_1, m_2, ..., m_n\}$ และ
ความน่าจะเป็นที่จะเกิดค่า $m_i = P(m_i)$ จะได้ว่า
จำนวนบิตน้อยที่สุดที่ใช้ encode m_i แต่ละตัว ที่ให้ค่าเฉลี่ยจำนวนบิตที่น้อยที่สุด คือ

Optimal code length $(m_i) = -\log_2 P(m_i)$

ค่าสารสนเทศของ M หรือค่าเอนโทรปีของ M เขียนแทนด้วย $I(M)$ คำนวณโดย

$$I(M) = \sum_{i}^{n} -P(m_i)\log_2 P(m_i)$$

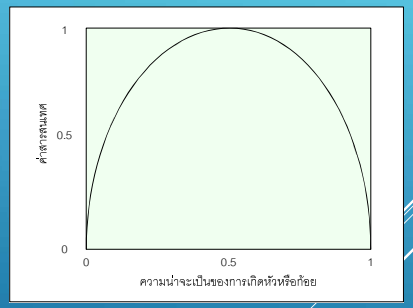
Message m_i	Probability	Standard Code	Optimal Code
A	0.5	00	0
B	0.25	01	10
C	0.125	10	110
D	0.125	11	111
Average Encoding Length		2 bits	1.75 bits

Average Encoding Length of Optimal Code is calculated by
=(-0.5*log0.5) +(-0.25*log0.25)+(-0.125*log0.125)+(-0.25*log0.125)
=(0.5*1)+(0.25*2)+(0.125*3)+(0.125*3)=1.75 bits

INFORMATION/ ENTROPY OF COIN FLIP

$$I(\text{การโยนหัวโยนก้อย}) = -P(\text{หัว})\log_2(P(\text{หัว})) - P(\text{ก้อย})\log_2(P(\text{ก้อย}))$$

- M1=HHHHHHHH
  I(M1) = (-1log$_2$1)+ (-0log$_2$ 0) = 0
- M2=TTTTTTTT
  I(M2) = (-0log$_2$ 0)+ (-1log$_2$ 1)= 0
- M3=HHHHTTTT
  I(M3) = (-0.5log$_2$ 0.5)+ (-0.5log$_2$ 0.5)= 1



Lower entropy implies a purer dataset. In a perfect case where the dataset contains only one class, the entropy is –(1*log$_2$1+0)=0

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

ID3 [J.R. QUINLAN]

Id3Estimator

ID3 CONSTRUCTION (1)

S=[9+, 5-]

$$I(S) = \frac{-9}{14}log_2\frac{9}{14} + \left(-\frac{5}{14}log_2\frac{5}{14}\right) = 0.940$$

Age

LE 30

S1=[2+, 3-]

$$I(S1) = \frac{-2}{5}log_2\frac{2}{5} + \left(-\frac{3}{5}log_2\frac{3}{5}\right)$$

[31..40]

S2=[4+, 0-]

$$I(S2) = \frac{-4}{4}log_2\frac{4}{4} + (-0\,log_2\,0)$$

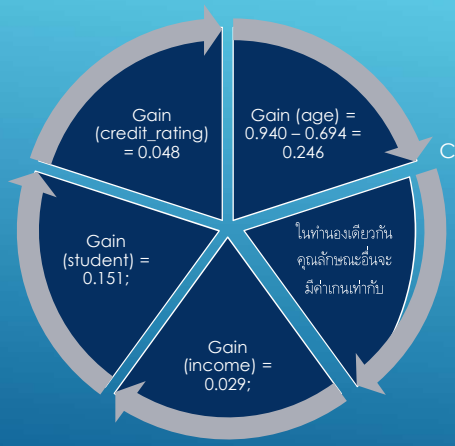
GT 40

S3=[3+, 2-]

$$I(S3) = -\frac{3}{5}log_2\frac{3}{5} + \left(-\frac{2}{5}log_2\frac{2}{5}\right)$$

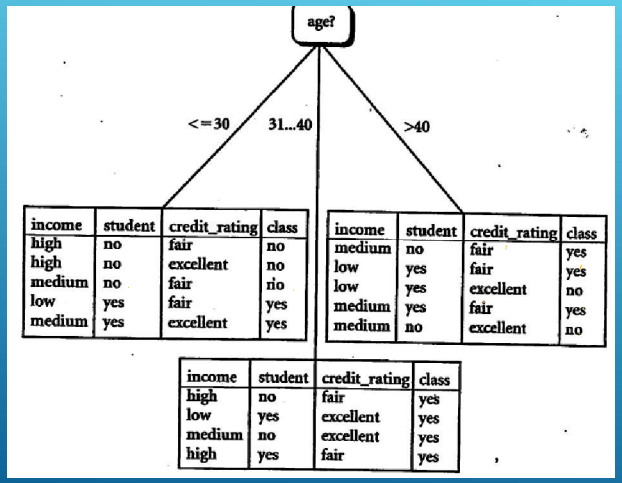
$$I_{age}(T) = \frac{5}{14}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}(0) + \frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694$$

ID3 CONSTRUCTION (2)



Choose attribute providing max GAIN → Age

ID3 CONSTRUCTION (3)

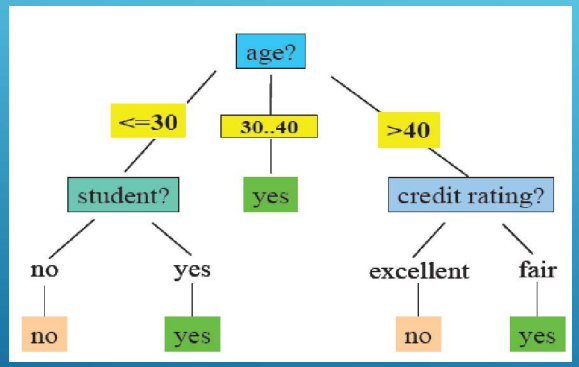


income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

OUTPUT OF ID3 LEARNING (MULTI-WAY SPLIT)



Gini Impurity

- Measure impurity rate of class distribution of data points
- Lower Gini indicates a purer dataset
- For a dataset with **K** classes, suppose data from class $k(1 \le k \le \mathbf{K})$, take up a fraction $f_k(0 \le f_k \le 1)$ of the entire dataset:

$$GiniImpurity = 1 - \sum_{k=1}^{K} f_k^2$$

- To evaluate quality of a split, add up the Gini of all resulting subgroups, combining the proportions of each subgroup as corresponding weight factors.
- The smaller the weighted sum of Gini Impurity, the better the split.

User gender	Interested in tech	Click	Group by gender
M	True	1	Group 1
F	False	0	Group 2
F	True	1	Group 2
M	False	0	Group 1
M	False	1	Group 1

#1 split based on gender

User gender	Interested in tech	Click	Group by interest
M	True	1	Group 1
F	False	0	Group 2
F	True	1	Group 1
M	False	0	Group 2
M	False	1	Group 2

#2 split based on interest in tech

Weighted Gini Impurity of #1split based on gender

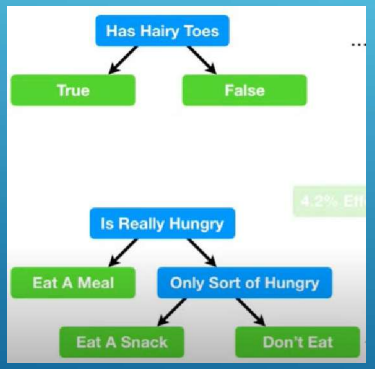
$$\#1\ Gini\ Impurity = \frac{3}{5}\left[1 - \left(\frac{2^2}{3} + \frac{1^2}{3}\right)\right] + \frac{2}{5}\left[1 - \left(\frac{1^2}{2} + \frac{1^2}{2}\right)\right] = 0.467$$

Weighted Gini Impurity of #2 split based on tech interest

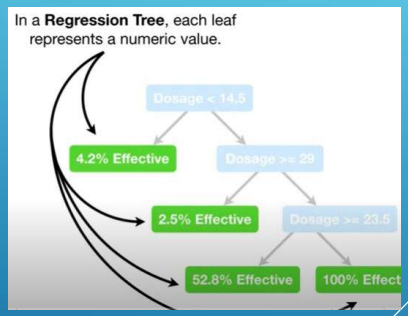
$$\#2\ Gini\ Impurity = \frac{2}{5}[1 - (1^2 + 0^2)] + \frac{3}{5}\left[1 - \left(\frac{1^2}{3} + \frac{2^2}{3}\right)\right] = 0.267$$

SPLIT METRICS: GINI INDEX

CART (Classification and Regression Tree)



Classification Tree



Regression Tree

- Most ML learning models in Python work with numerical data
- Three approaches to manage categorical data:
  - Drop categorical variables if NOT relevant
  - Label encoding or ranking in case of ordinal variables
  - One-Hot encoding

Label	Encoded Label
Africa	1
Asia	2
Europe	3
South America	4
North America	5
Other	6

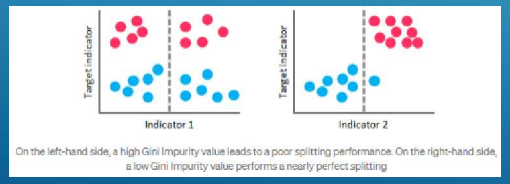
Label encoding

	is_africa	is_asia	is_europe	is_sam	is_nam
Africa	1	0	0	0	0
Asia	0	1	0	0	0
Europe	0	0	1	0	0
South America	0	0	0	1	0
North America	0	0	0	0	1
Other	0	0	0	0	0

One-hot encoding

DATA PREPROCESSING

- Always produces **binary** splits
- **Gini index**. A Gini score of 0 indicates perfect purity and a score of 1 indicates maximum impurity.
- CART should be allowed to go till 7–8 tree depth in accordance with the nature of producing tall and skinny trees.
- Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned).
- The optimal Tree is identified by evaluating the performance of every Tree through test set; or performing k-fold cross-validation.



On the left-hand side, a high Gini Impurity value leads to a poor splitting performance. On the right-hand side, a low Gini Impurity value performs a nearly perfect splitting

CART ALGORITHM

K-Fold CV

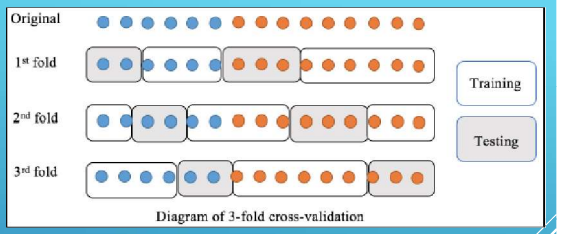
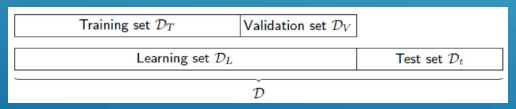


Diagram of 3-fold cross-validation

Holdout



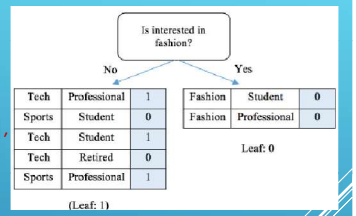
MODEL ASSESSMENT

- Gini(interest, tech) = weighted_impurity([[1, 1, 0], [0, 0, 0, 1]]) = 0.405
- Gini(interest, Fashion) = weighted_impurity([[0, 0], [1, 0, 1, 0, 1]]) = 0.343
- Gini(interest, Sports) = weighted_impurity([[0, 1], [1, 0, 0, 1, 0]]) = 0.486
- Gini(occupation, professional) = weighted_impurity([[0, 0, 1, 0], [1, 0, 1]]) = 0.405
- Gini(occupation, student) = weighted_impurity([[1, 0, 0, 1], [0, 0, 1]]) = 0.476
- Gini(occupation, retired) = weighted_impurity([[1, 0, 0, 0, 1, 1], [0]]) = 0.429

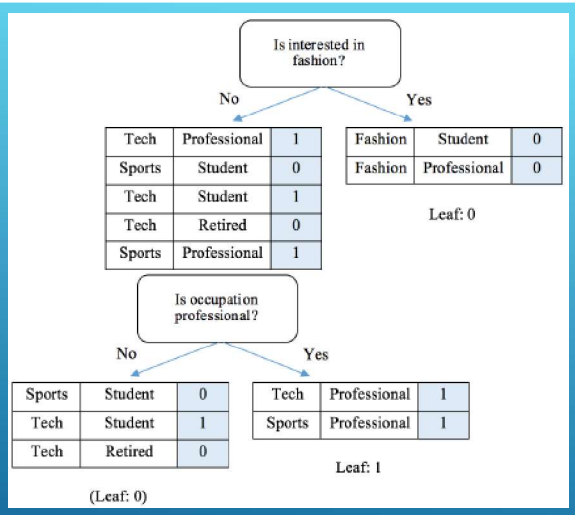
User interest	User occupation	Click
Tech	Professional	1
Fashion	Student	0
Fashion	Professional	0
Sports	Student	0
Tech	Student	1
Tech	Retired	0
Sports	Professional	1

IMPLEMENTING A CART TREE (1)

- Gini(interest, tech) = weighted_impurity([[0, 1], [1, 1, 0]]) = 0.467
- Gini(interest, Sports) = weighted_impurity([[1, 1, 0], [0, 1]]) = 0.467
- Gini(occupation, professional) = weighted_impurity([[0, 1, 0], [1, 1]]) = 0.267
- Gini(occupation, student) = weighted_impurity([[1, 0, 1], [0, 1]]) = 0.467
- Gini(occupation, retired) = weighted_impurity([[1, 0, 1, 1], [0]]) = 0.300



IMPLEMENTING A CART TREE (2)



IMPLEMENTING A CART TREE (3)

Weight	Heart Disease
220	Yes
180	Yes
225	Yes
190	No
155	No

	Weight	Heart Disease
Lowest	155	No
	180	Yes
	190	No
	220	Yes
Highest	225	Yes

Step 1) Sort the patients by weight, lowest to highest.

NUMERIC VARIABLE: WHAT'S THE BEST WEIGHT USED TO DIVIDE THE PATIENT?

Step 2) Calculate the average weight for all adjacent patients.

Step 3) Calculate the impurity values for each average weight.

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

Weight	Heart Disease	
155	No	
167.5	Yes	Gini impurity = 0.3
180		
185	No	Gini impurity = 0.47
190		
205	Yes	Gini impurity = 0.27
220		
222.5	Yes	Gini impurity = 0.4
225		

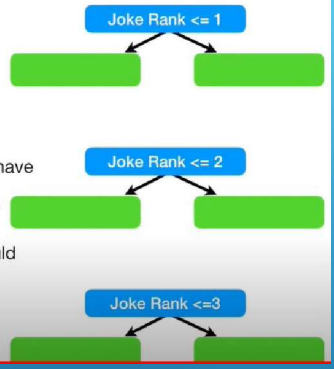
The lowest impurity occurs when we separate using **weight < 205**…

…so this is the cutoff and impurity value we will use when we compare weight to chest pain or blocked arteries.

NUMERIC VARIABLE: WHAT'S THE BEST WEIGHT USED TO DIVIDE THE PATIENT?

Rank my jokes…	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc…	etc…

NOTE: We don't have to calculate an impurity score for Joke Rank <= 4 because that would include everyone.



ORDINAL VARIABLE

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc…	etc…

When there are **multiple choices**, like "**color choice can be blue, green or red**", you calculate an impurity score for each one as well as each possible combination.

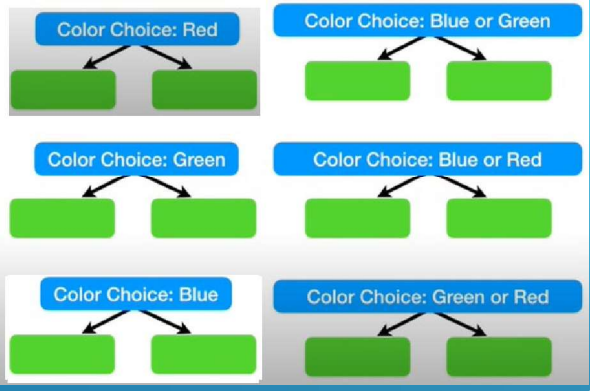
NOMINAL VARIABLE (1)

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc…	etc…

1

2

3



NOTE: We don't have to calculate an impurity score for "Color Choice: Blue or Green or Red" since that includes everyone.

NOMINAL VARIABLE (2)

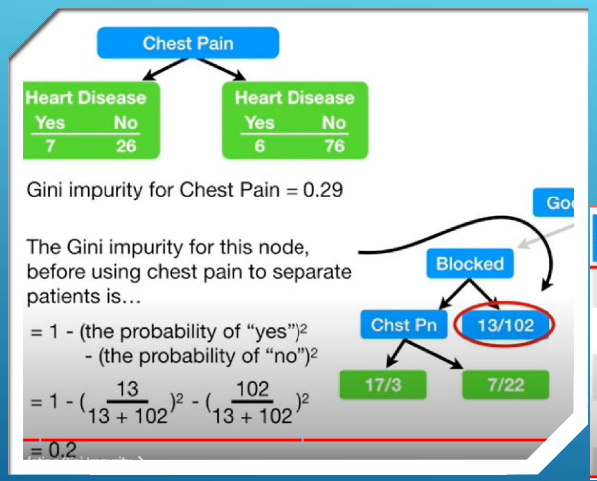
- Pruning is a technique used to deal with overfitting, that reduces the size of DTs by removing sections of the Tree that provide little predictive or classification power.
- The goal is to reduce complexity and gain better accuracy by reducing the effects of overfitting and removing sections of the DT that may be based on noisy or erroneous data.
- The pruned model is less complex, explainable, and easy to understand than the unpruned.
- Two different strategies to perform pruning on DTs:
  - Pre-pruning
  - Post-pruning

TREE PRUNING

- **Prepruning** is the halting of subtree construction at some node after checking some measures: Gini.
- If partitioning the tuple at a node would result in a split that falls below a prespecified threshold, then pruning is done.
- Pre-pruning may stop the growth process prematurely.

- **Postpruning** grows decision tree to its entirety.
- Trim the nodes of DT in a *bottom-up* fashion.
- Postpruning is done by replacing the node with leaf.
- If error improves after trimming, replace sub- tree by a leaf node.

PREPRUNING VS. POSTPRUNING

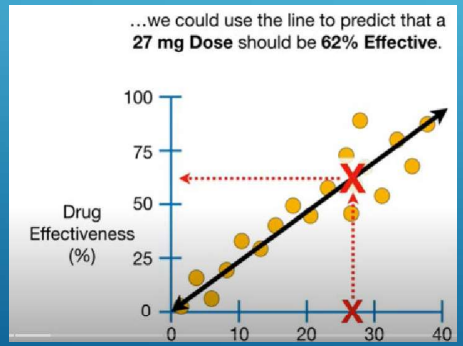


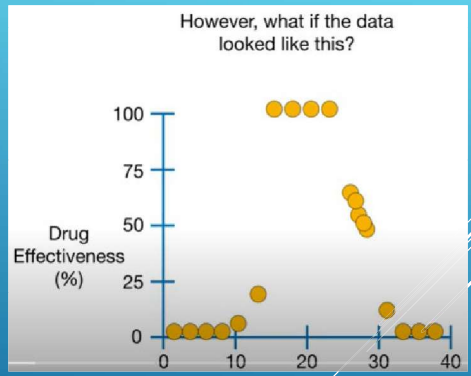
STOP SPLITTING WHEN NO FURTHER GAIN CAN BE MADE

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc…	etc…	etc…	etc…

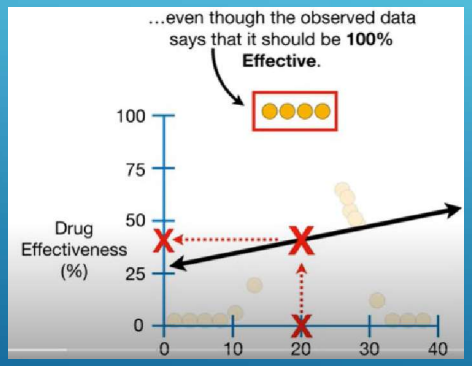
LINEAR REGRESSION

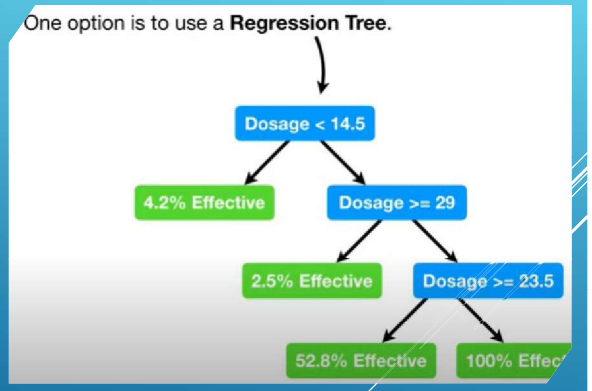
Easily fit a line to the data, the higher the dose, the more effective the drug…

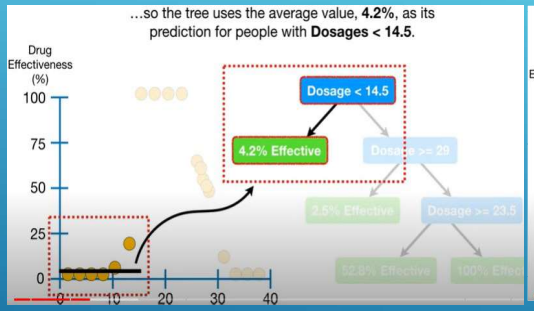


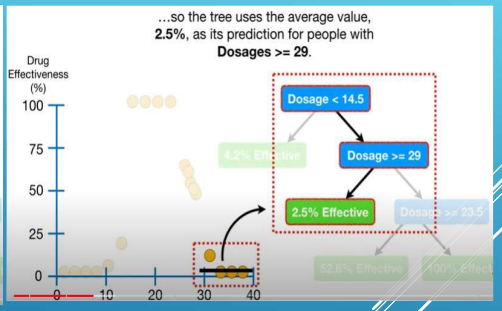


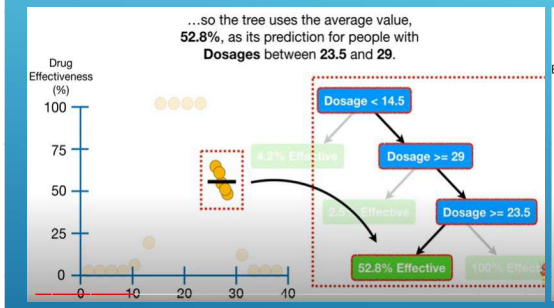
For example, if someone told us they were taking a 20 mg Dose…

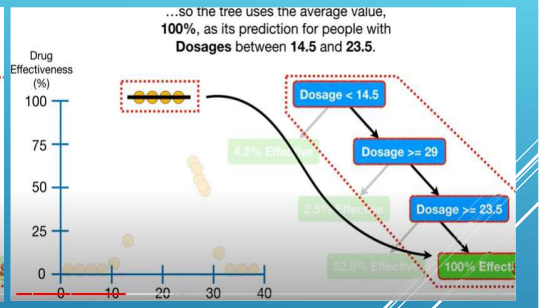


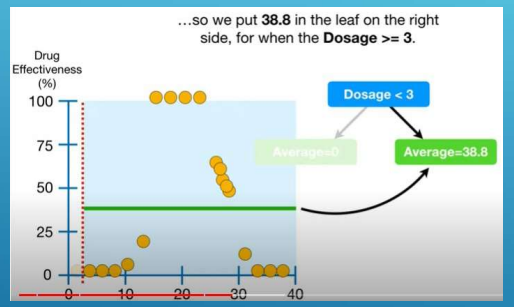


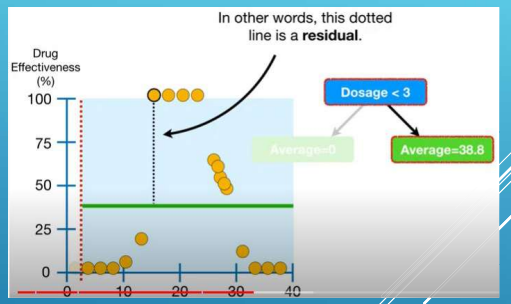


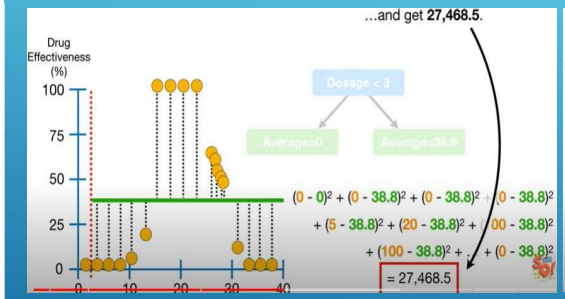


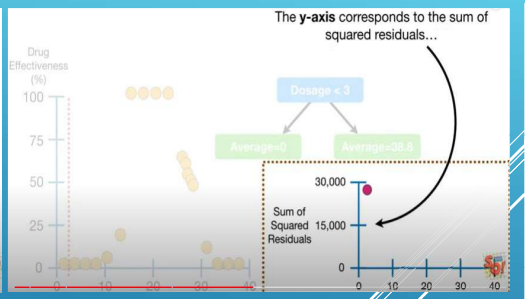


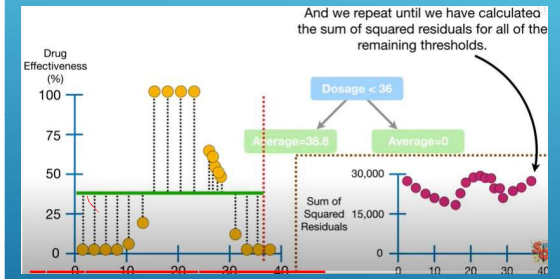


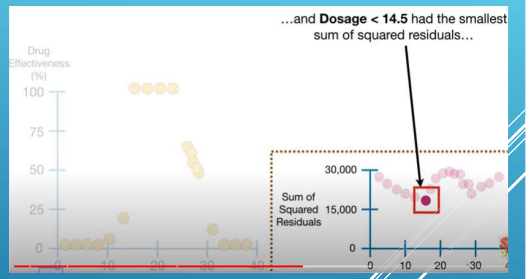


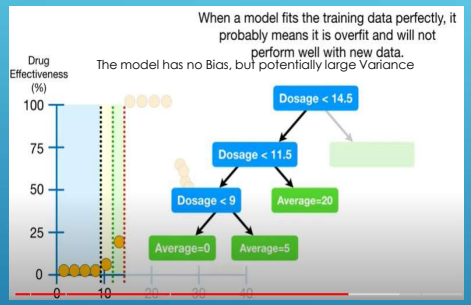


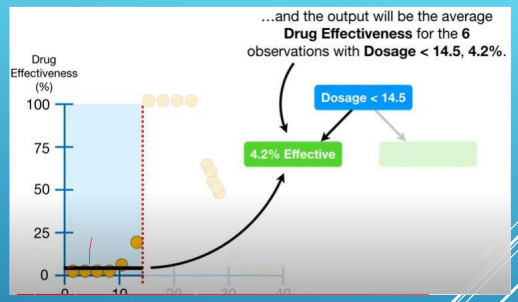






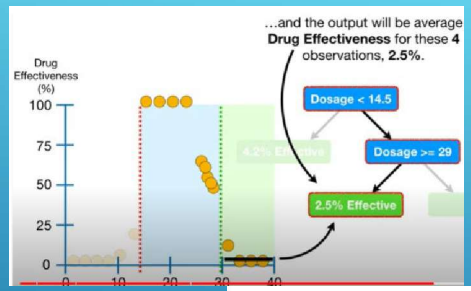


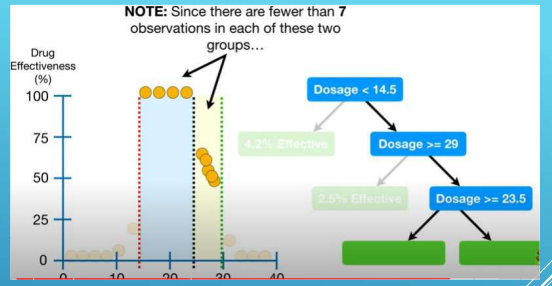


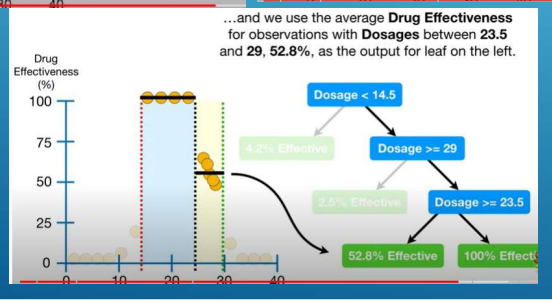


The simplest is to only split observations when there are more than some minimum number. Typically, the minimum number of observations to allow for a split is **20**.

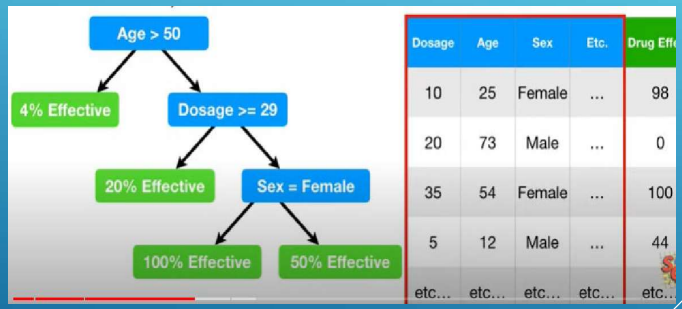
However, since this example doesn't have many observations, I set the minimum to **7**.





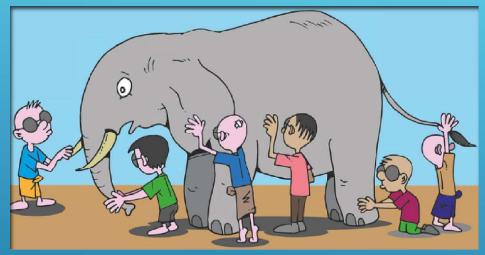


- A Regression tree looks for splits that minimize the Least Square Deviation (LSD), sometimes referred as "variance reduction", that implies the variance within the node.



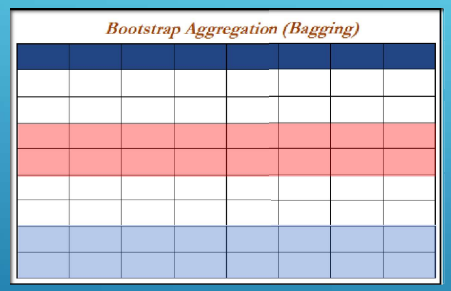
Dosage	Age	Sex	Etc.	Drug Eff
10	25	Female	...	98
20	73	Male	...	0
35	54	Female	...	100
5	12	Male	...	44
etc...	etc...	etc...	etc...	etc...

REGRESSION TREE



- Combine decisions from multiple models to improve overall performance
- Help minimize causes of error due to noise, bias and variance
- Major schemes:
  - Bagging
  - Boosting

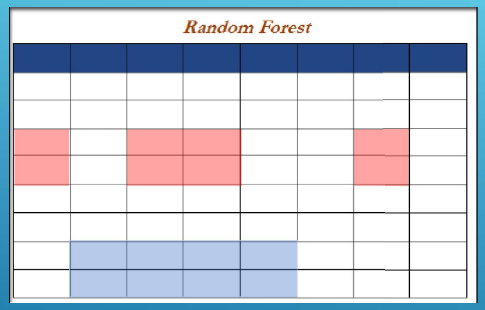
PART2: DECISION TREE ENSEMBLES



Two samples (pink, blue) with all variables

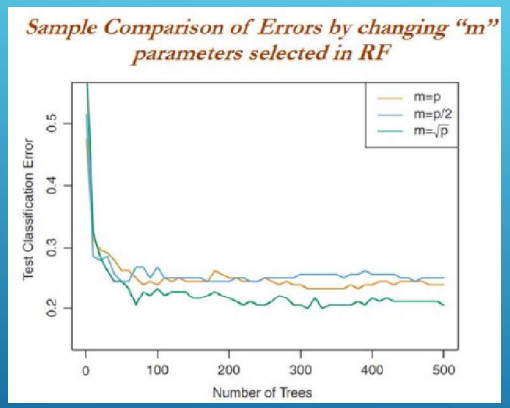
- **Bootstrap aggregation** or **bagging** introduced by Leo Breiman in 1994.
- Bootstrapped datasets are created by **sampling with replacement**.
- Build a number of decision trees on bootstrapped samples from training data
- Combine the results of the models by **averaging** or **majority voting**
- The algorithm aims to reduce the chance of overfitting.
- Due to all variables selected, order of candidate/variable chosen to split remains more or less the same for all the individual trees.
- Variance reduction on correlated individual entities does not work effectively while aggregating them.

BAGGING



- In bagging, a random sample with replacement is selected to train every tree in the ensemble. The tree is trained on all the features.
- While each decision tree in the random forest is given a randomly selected subset of features and a randomly selected subset of the dataset for the selected features.

RANDOM FORESTS



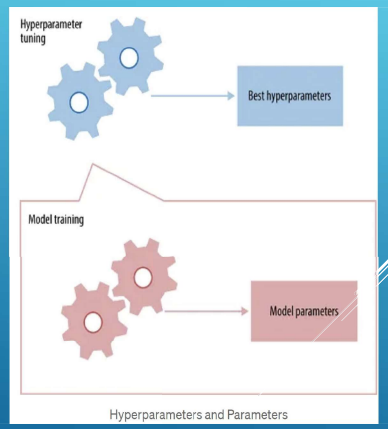
Bagging → m=p
RF (Classification) → m = sqrt(p)
RF (Regression) → m = p/2

Though thumb rules suggest selecting m variables out of the total p, it is encouraged to tune the number of variables/ predictors

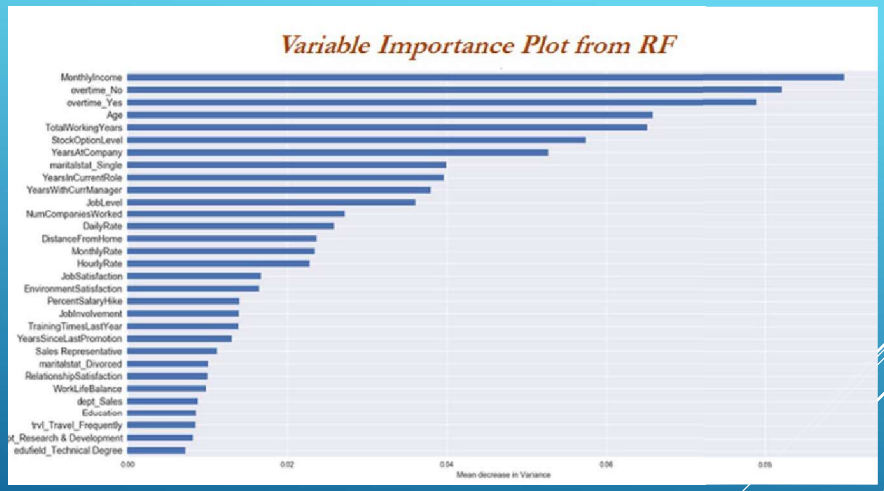
HYPERPARAMETERS IN RANDOM FOREST

- max_depth: the deepest individual tree. It tends to overfit if it is too deep, or to underfit if it is too shallow.
- min_samples_split: represents min number of samples required for further splitting at a node. Too small a value tends to cause overfitting, while too large a value is likely to introduce underfitting. 10, 30, and 50 might be good options to start with.
- n_estimators: represents the number of trees considered for majority voting. The more trees, the better the performance, but more computation time. It is usually set as 100, 200, 500, and so on.
- Practically, application of **grid search** on tuning different combinations of hyperparameters will provide better and more robust results.

- model ***parameters*** are learned during training, e.g. *slope* and *intercept* in a linear regression
- ***hyperparameters*** must be set by the data scientist **before** training



Hyperparameters and Parameters



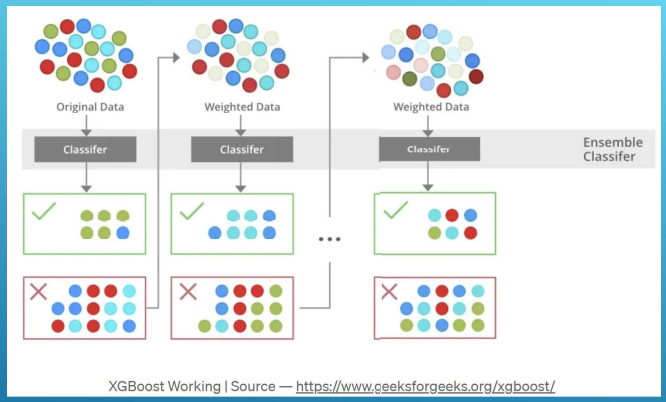
- In boosting, all models are trained in sequence, instead of in parallel as in bagging.
- The weights are reassigned after a model is trained, which will be used for the next training round.
- In general, weights for mispredicted samples are increased to stress their prediction difficulty.
- There are many boosting algorithms e.g. AdaBoost, Gradient Boosting and XGBoost; boosting algorithms differ mostly in their weighting scheme.
- Boosting relies on creating a series of weak learners each of which might not be good for the entire data set but is good for some part of the data set. Thus, each model actually boosts the performance of the ensemble.
- Boosting has shown better predictive accuracy than bagging.

BOOSTING

XGBOOST

- XGBoost is a gradient boosting algorithm, originally developed by *Tianqi Chen*.
- It works by combining a number of weak learners to form a strong learner.
- XGBoost works by training a number of decision trees. Each tree is trained on a subset of the data, and the predictions from each tree are combined to form the final prediction.
- A number of most important parameters include:

  - ***max_depth:*** The maximum depth of the decision trees.
  - ***eta:*** The learning rate.
  - ***gamma:*** The minimum loss reduction required to make a split.
  - ***subsample:*** The fraction of the training data that is used to train each tree.



XGBoost Working | Source — https://www.geeksforgeeks.org/xgboost/

XGBoost has been shown to outperform other machine learning algorithms in a variety of tasks, including classification, regression and ranking.

DIFFERENCES BETWEEN BAGGING AND BOOSTING

	Bagging	Boosting
Differences	Individual models are built separately	Each new model is influenced by the performance of those built previously
	Equal weight is given to all models	Weights a model's contribution by its performance

- DT can be applied to either classification or regression problems.
- Able to handle both numerical and categorical variables.
- No assumptions are made on the underlying distribution of the data.
- Useful in data exploration: DT is one of the fastest ways to identify the most significant variables.
- White box type algorithm– it shares internal decision-making logic, viz not avail in black box type e.g. Neural Network (NN)
- Faster training time compared to NN

- Overfitting/ high variance error is one of the most practical difficulties for DT models. The problem can be solved by pruning and ensemble techniques.