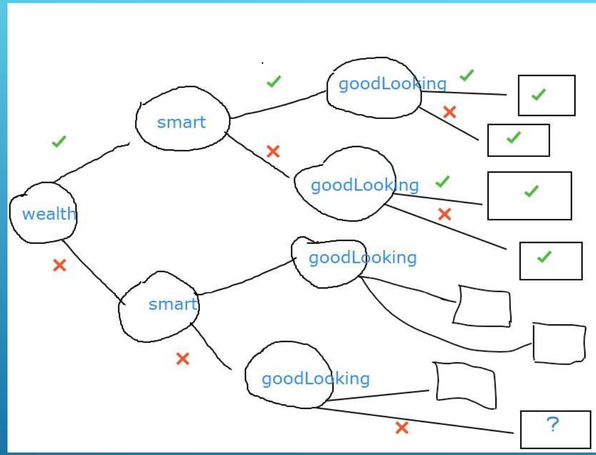
CHAPTER 7
DECISION TREE

PART I: DECISION TREE CLASSIFIERS
PART II: DECISION TREE ENSEMBLES

Associate Professor Yachai Limpiyakorn, Ph.D

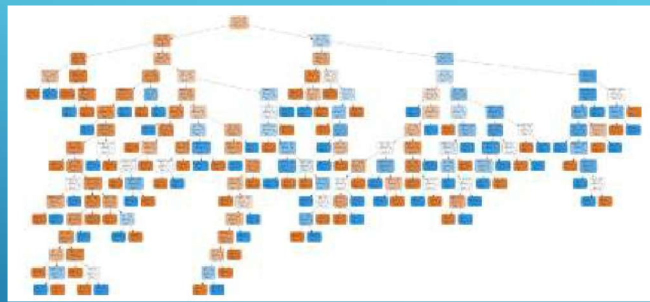


GENERAL DECISION MAKING TREE

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

ประเภท

คุณสมบัติ

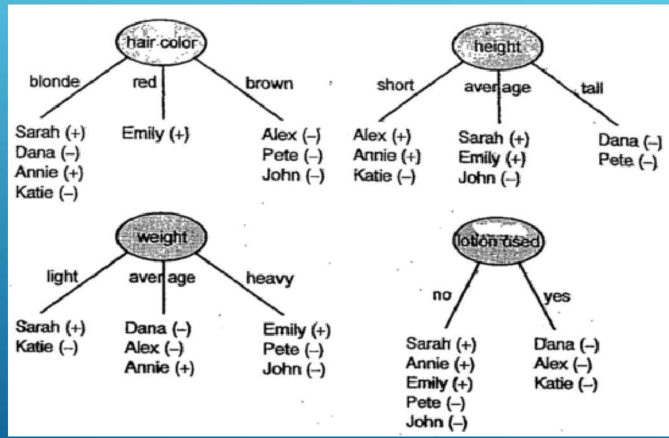
Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned
Dana	blonde	tall	average	yes	none
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

ค่า

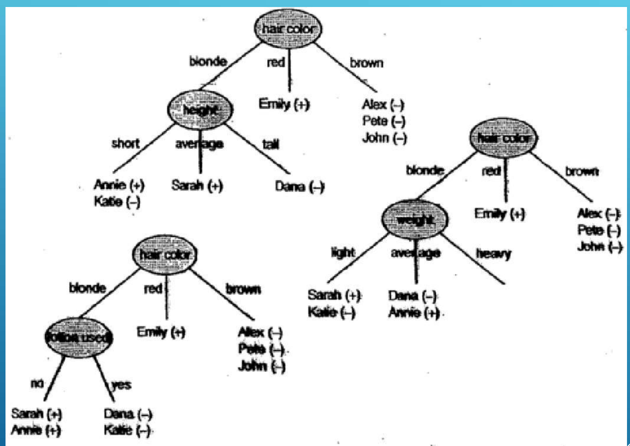
- สร้างต้นไม้จำนวนเท่ากับค่าคุณลักษณะ และให้แต่ละคุณลักษณะเป็นโหนดราก
- สำหรับต้นไม้แต่ละต้น แบ่งตัวอย่างสอนไปแต่ละกิ่งตามค่าคุณลักษณะของโหนดราก
- สำหรับต้นไม้แต่ละต้น ให้สร้างโหนดภายในด้วยคุณลักษณะที่เหลือ
- ดังตัวอย่างการสร้างต้นไม้ที่มี "hair color" เป็นโหนดราก
- วนทำเช่นนี้ไปเรื่อยๆ จนกระทั่งสร้างต้นไม้ตัดสินใจที่เป็นไปได้ครบหมดทุกต้น

OPTIMAL DECISION TREE

สร้างต้นไม้ที่ประกอบด้วยโหนดราก 4 แบบที่เป็นไปได้



โหนดราก HAIR COLOR และโหนดภายใน 3 แบบที่เป็นไปได้



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	Umbrella
1	1	0	Rain jacket
1	1	1	Stay home

TREE CONSTRUCTION

Occam's Razor

สมมติฐานที่สั้นกว่าที่สามารถอธิบายข้อมูลได้เหมือนกัน จะเป็นสมมติฐานที่ดีกว่า

the simplest explanation is usually the best one

~raining → don't bring anything

raining and ~windy → use an umbrella

raining and ~extremely windy → wear a rain jacket

raining and extremely windy → stay home

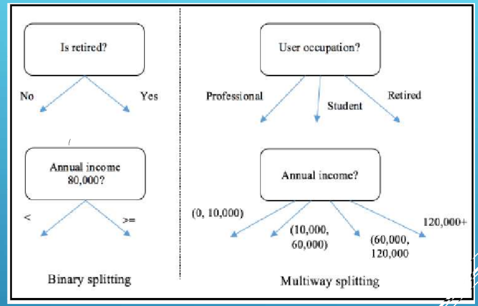
https://medium.com/@ml.at.berkeley/machine-learning-crash-course-part-5-decision-trees-and-ensemble-models-dcc5a36af8cd

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:

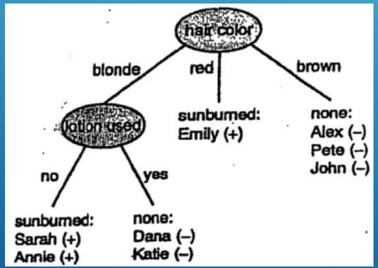
- All the tuples belong to the same class.
- There are no more remaining attributes.
- There are no more instances.



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)

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

SPLIT MEASURE/ ASM

กำหนด message M ประกอบด้วยค่าที่เป็นไปได้ $\{m_1, m_2, \ldots, m_n\}$ 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	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

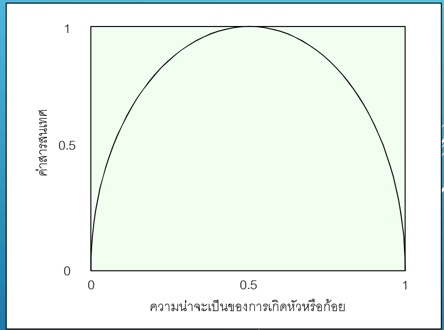
ENTROPY/ INFORMATION

INFORMATION/ ENTROPY OF COIN FLIP

$\log_2 1 = 0$
$2^0 = 1$

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

- M1=HHHHHHHH
  I(M1) = (-1log$_2$1)+ (-0log$_2$0) = 0 ϕ
- M2=TTTTTTTT
  I(M2) = (-0log$_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*log₂1+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 | [31..40] | GT 40

S1=[2+, 3-]

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

S2=[4+, 0-]

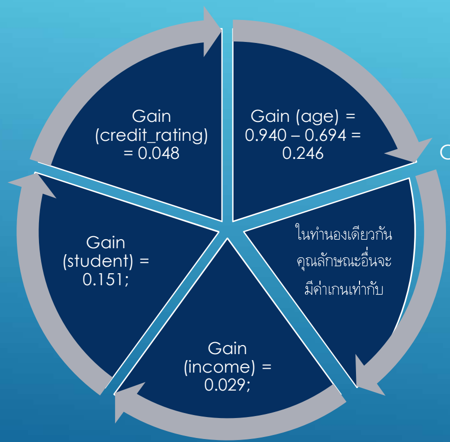
I(S2)= $\frac{-4}{4}log_2\frac{4}{4} + (-0\, log_2\, 0)$

S3=[3+, 2-]

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

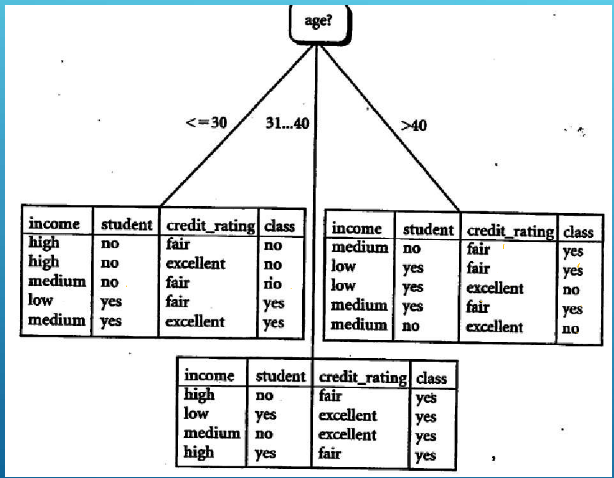
$$I_{age}(T) = \frac{5}{14}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}(0) + \frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694$$

ID3 CONSTRUCTION (2)



Choose attribute providing max GAIN → Age

ID3 CONSTRUCTION (3)

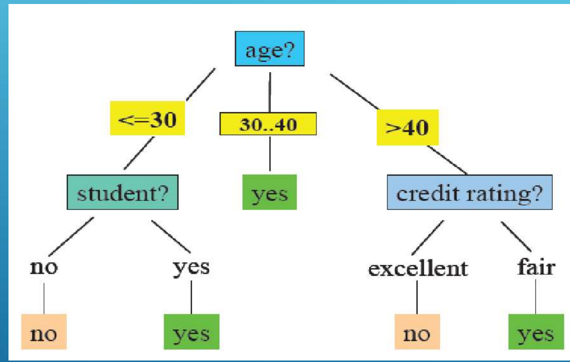


income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

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

OUTPUT OF ID3 LEARNING (MULTI-WAY SPLIT)



SPLIT METRICS: GINI INDEX

Gini Impurity

- Lower Gini indicates a purer dataset
- For a dataset with **K** classes, suppose data from class $k (1 \le k \le K)$, 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

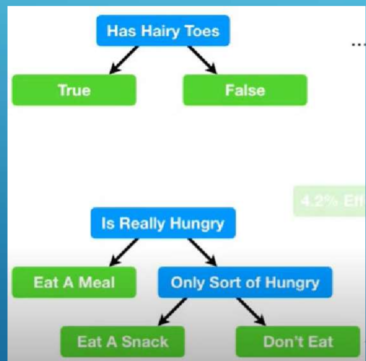
Weighted Gini Impurity of #1split based on gender

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

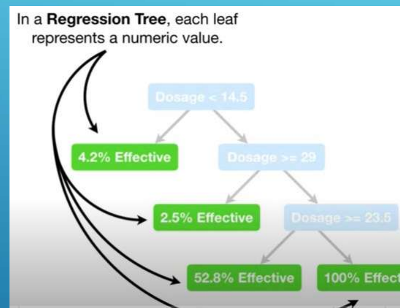
Weighted Gini Impurity of #2 split based on tech interest

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

CART (Classification and Regression Tree)



Classification Tree



Regression Tree

DATA PREPROCESSING

- 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 encoding

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

	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

ONE-HOT ENCODING

ข้อดี

- ช่วยให้การเรียนรู้ง่ายขึ้น แทนที่โมเดลจะเรียนรู้ Pattern เปลี่ยนมาเรียนรู้จากสวิตช์
- ความหมายของข้อมูลแบบ Nominal จะตรงขึ้น Europe 3 ไม่ได้ใกล้เคียง S.America 4 มากกว่า N. America 5
- สามารถ Dot Product กับ Matrix ที่ต้องการ

ข้อเสีย

- ความหมายของลำดับข้อมูลแบบ Ordinal จะหายไป เนื่องจากทุก Category แตกต่างกันเท่ากันหมด
- ถ้าข้อมูลมี Value หลากหลายมาก เช่น มีสีเสื้อ 10,000 สี จะทำให้มีปัญหาเปลืองเนื้อที่/Memory ที่เก็บค่า 0 เป็นส่วนใหญ่ เรียกว่า Sparse Matrix
- การเพิ่ม Categoryใหม่ ยิบย่อยอยู่ตลอด จะทำให้มีปัญหา เช่น เพิ่มสีเสื้อใหม่

CART ALGORITHM

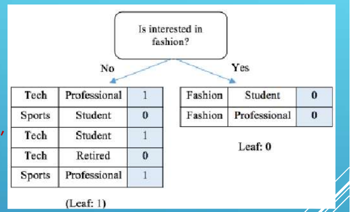
- Always produces **binary** splits
- **Gini index**. A Gini score of 0 indicates perfect purity and a score of 1 indicates maximum impurity.
- CART should be allowed to go till 7–8 tree depth in accordance with the nature of producing tall and skinny trees.
- Splitting stops when CART detects no further 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.

IMPLEMENTING A CART TREE (1)

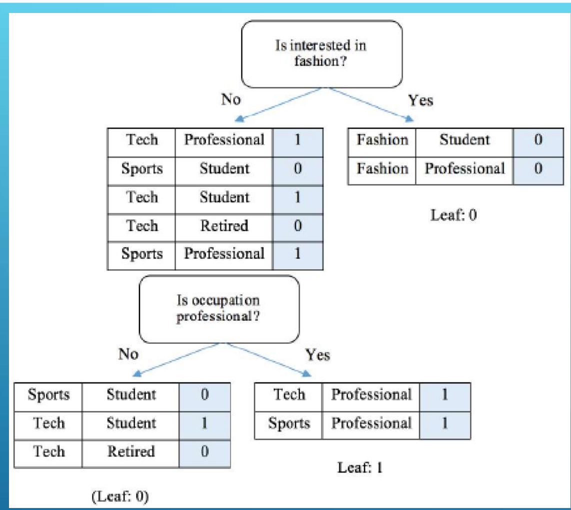
- Gini(interest, tech) = weighted_impurity([[1, 1, 0], [0, 0, 0, 1]]) = 0.405
- Gini(interest, Fashion) = weighted_impurity([[0, 0], [1, 0, 1, 0, 1]]) = 0.343
- Gini(interest, Sports) = weighted_impurity([[0, 1], [1, 0, 0, 1, 0]]) = 0.486
- Gini(occupation, professional) = weighted_impurity([[0, 0, 1,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

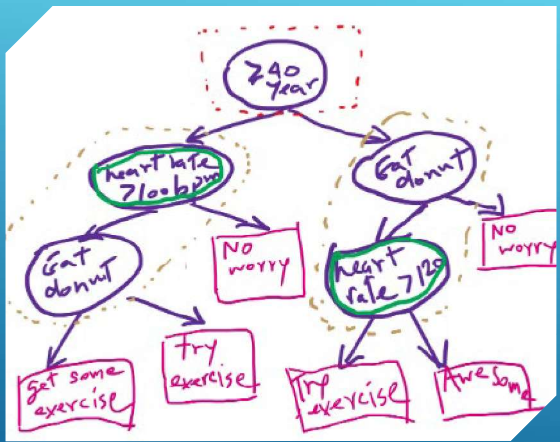
- Gini(interest, tech) = weighted_impurity([[0, 1], [1, 1, 0]]) = 0.467
- Gini(interest, Sports) = weighted_impurity([[1, 1, 0], [0, 1]]) = 0.467
- Gini(occupation, professional) = weighted_impurity([[0, 1, 0], [1, 1]]) = 0.267
- Gini(occupation, student) = weighted_impurity([[1, 0, 1], [0, 1]]) = 0.467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



IMPLEMENTING A CART TREE (2)



IMPLEMENTING A CART TREE (3)



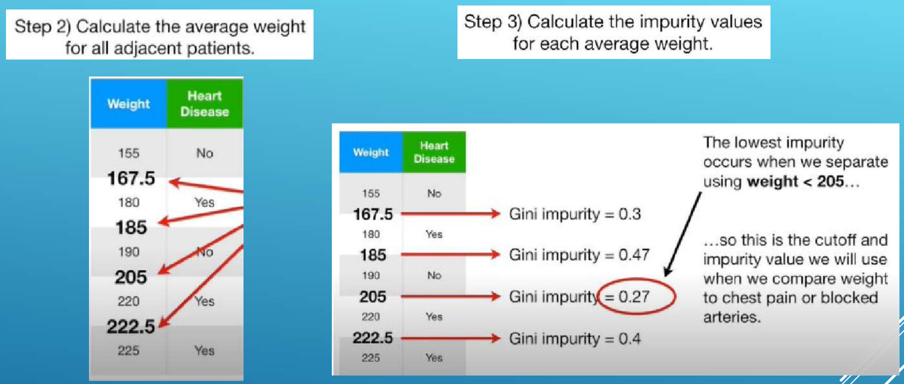
EXPLORATION

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

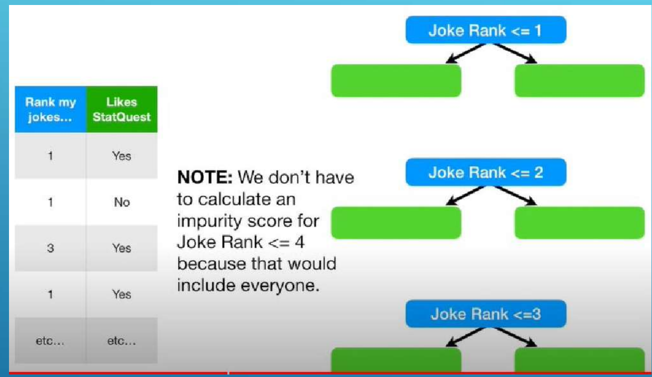
	Weight	Heart Disease
Lowest	155	No
	180	Yes
	190	No
	220	Yes
Highest	225	Yes

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

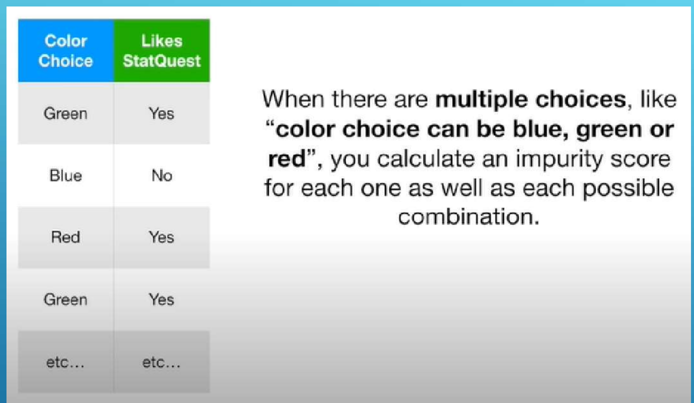
NUMERIC VARIABLE: WHAT'S THE BEST WEIGHT USED TO DIVIDE THE PATIENT?



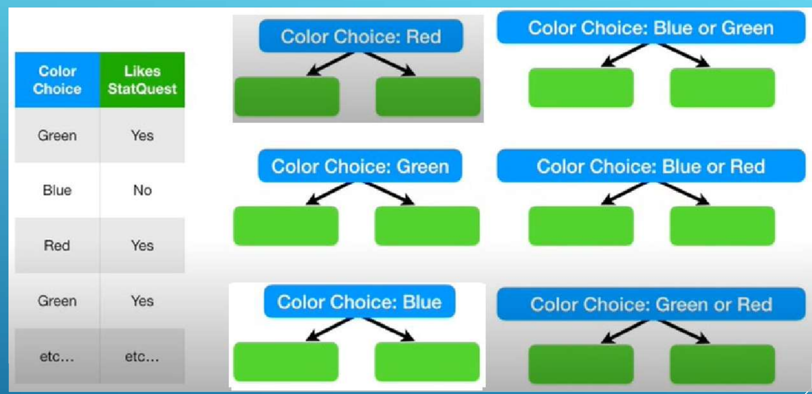
NUMERIC VARIABLE: WHAT'S THE BEST WEIGHT USED TO DIVIDE THE PATIENT?



ORDINAL VARIABLE



NOMINAL VARIABLE (1)

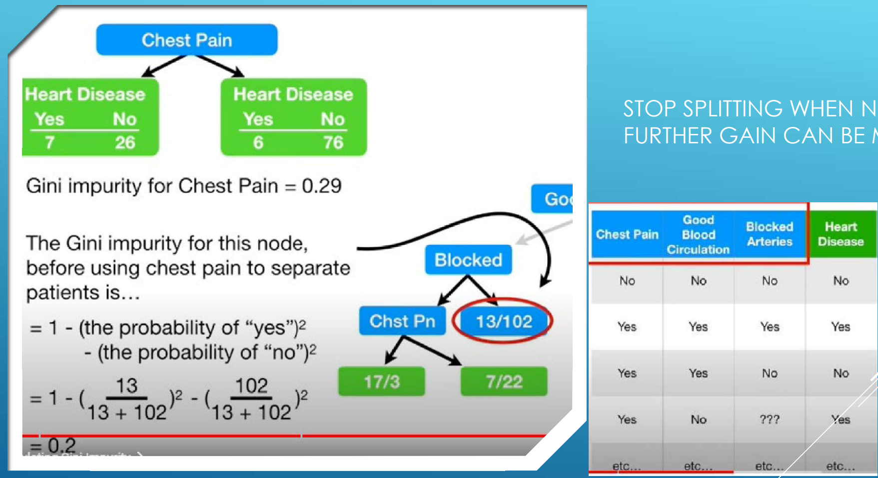


NOTE: We don't have to calculate an impurity score for "Color Choice: Blue or Green or Red" since that

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.
- There are two different strategies to perform pruning on DTs:
  - Pre-prune: When you stop growing DT branches when information becomes unreliable.
  - Post-prune: When you take a fully grown DT and then remove leaf nodes only if it results in a better model performance. This way, you stop removing nodes when no further improvements can be made.

TREE PRUNING



STOP SPLITTING WHEN NO FURTHER GAIN CAN BE MADE

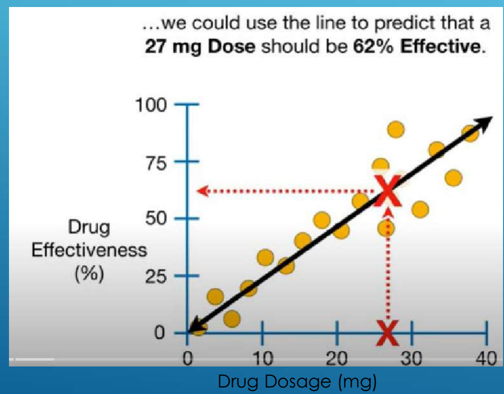
- Optimization of DT classifier performed by only pre-pruning using maximum depth of DT.
- **max_depth : int or None, (default=None) or Maximum Depth of a Tree**: If None, nodes are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting.

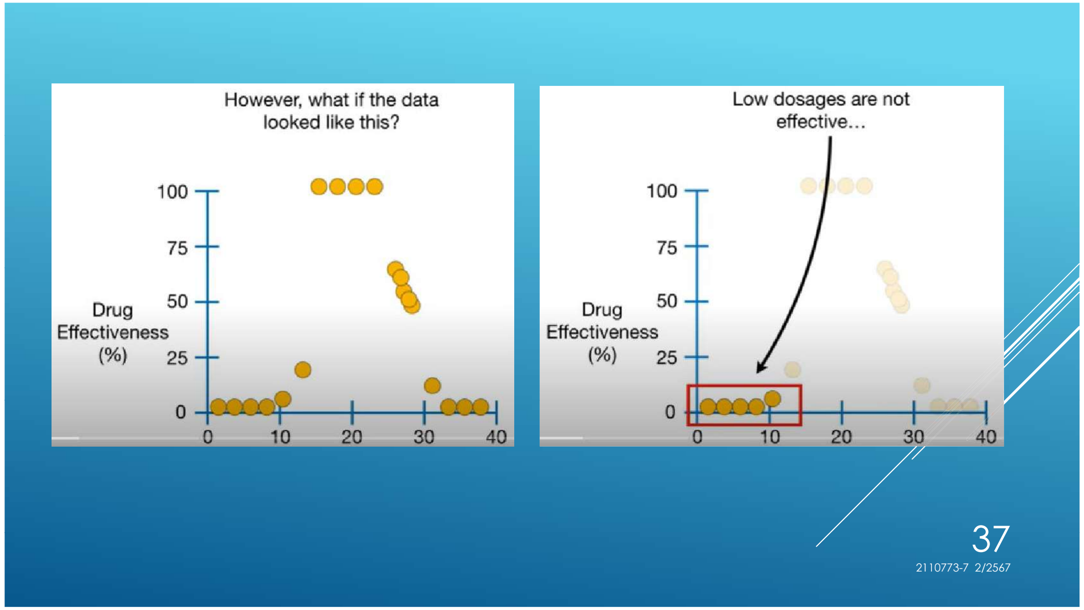
DECISION TREE (CLASSIFICATION) USING SCIKIT-LEARN

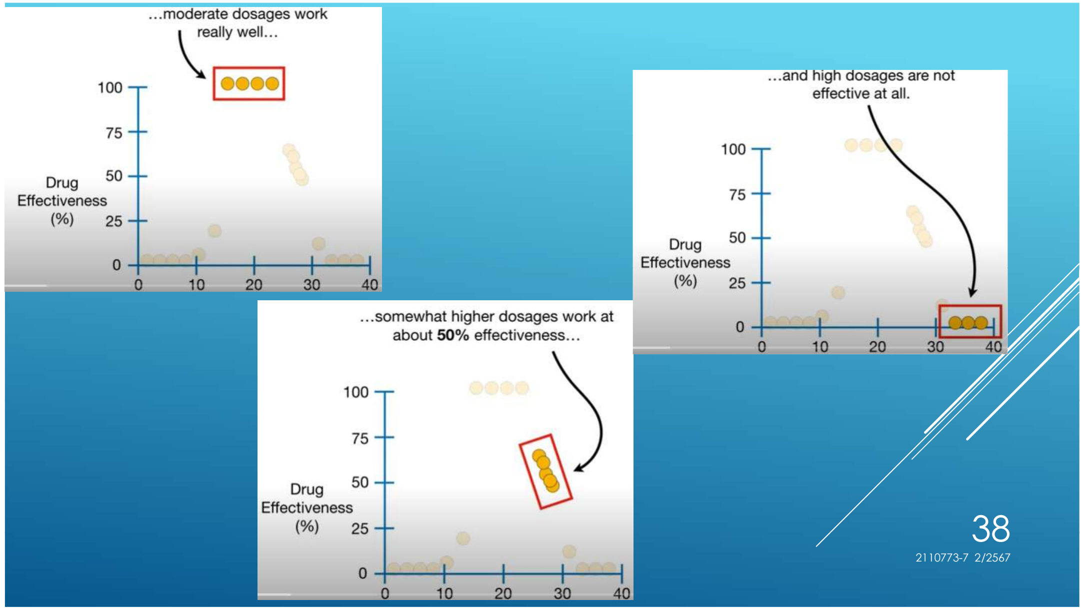
HTTPS://WWW.DATACAMP.COM/COMMUNITY/TUTORIALS/DECISION-TREE-CLASSIFICATION-PYTHON

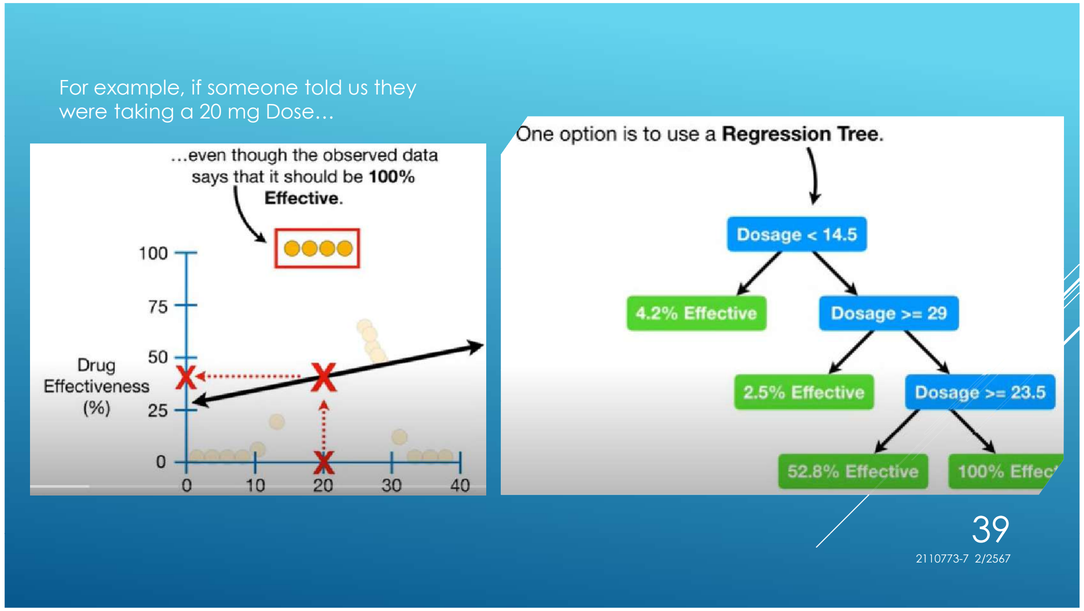
LINEAR REGRESSION

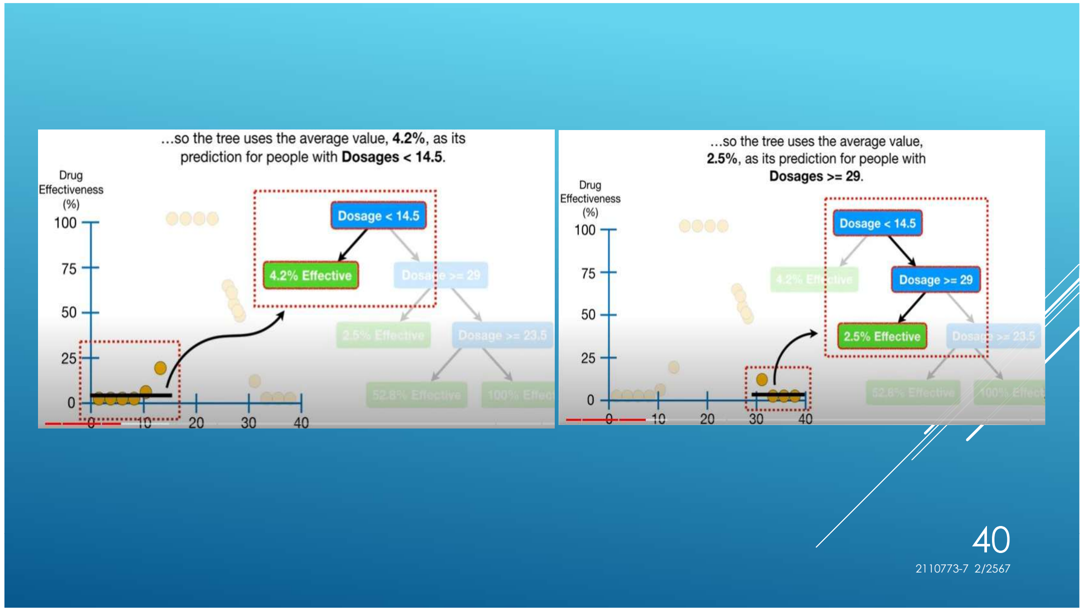
Easily fit a line to the data, the higher the dose, the more effective the drug…

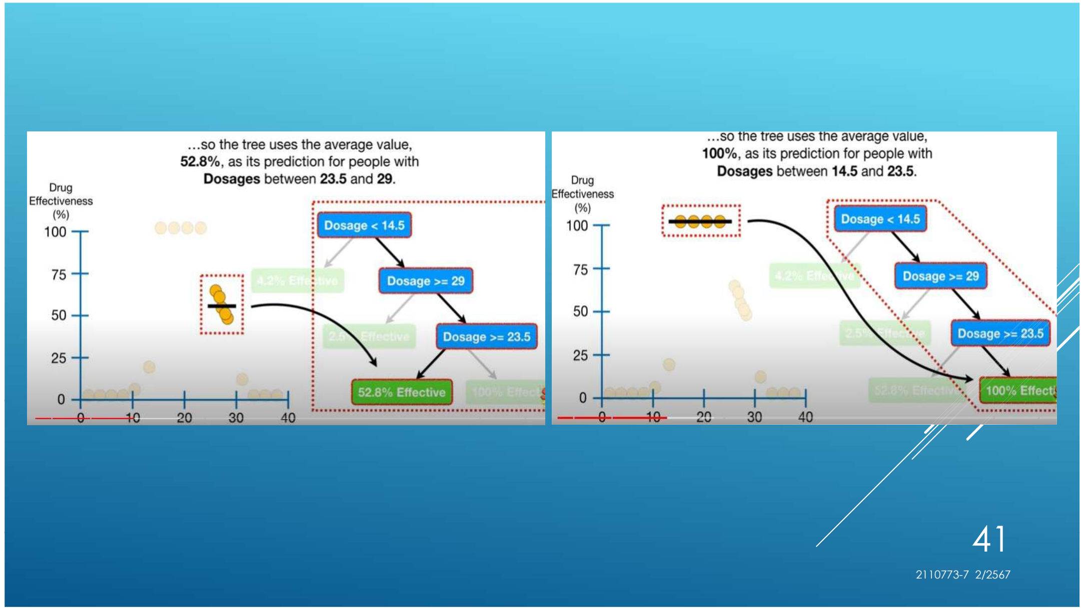


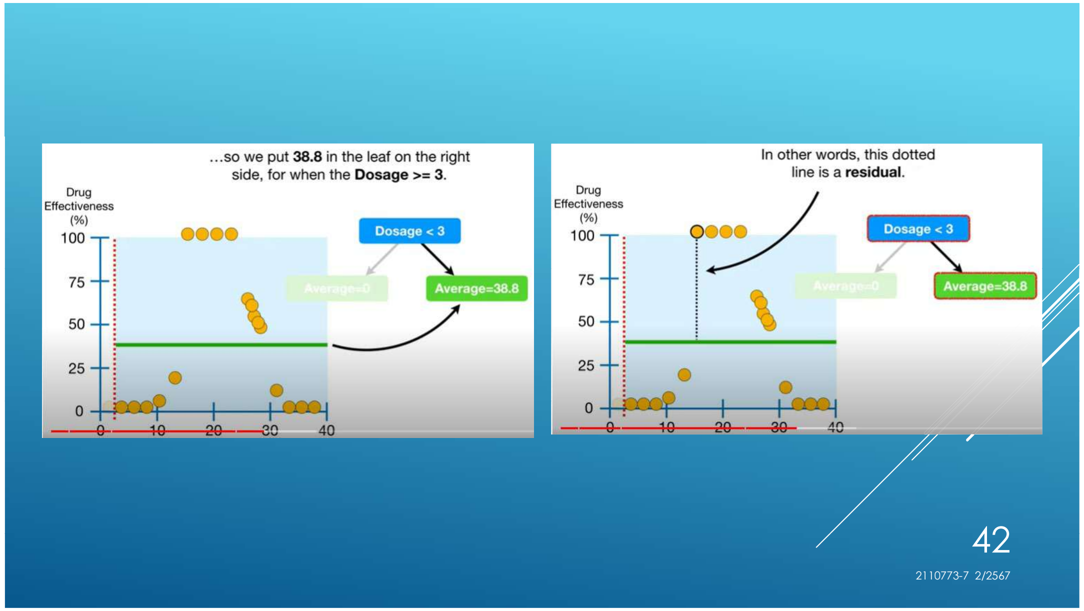

However, what if the data looked like this?
Drug Effectiveness (%)
Low dosages are not effective…
37
2110773-7 2/2567

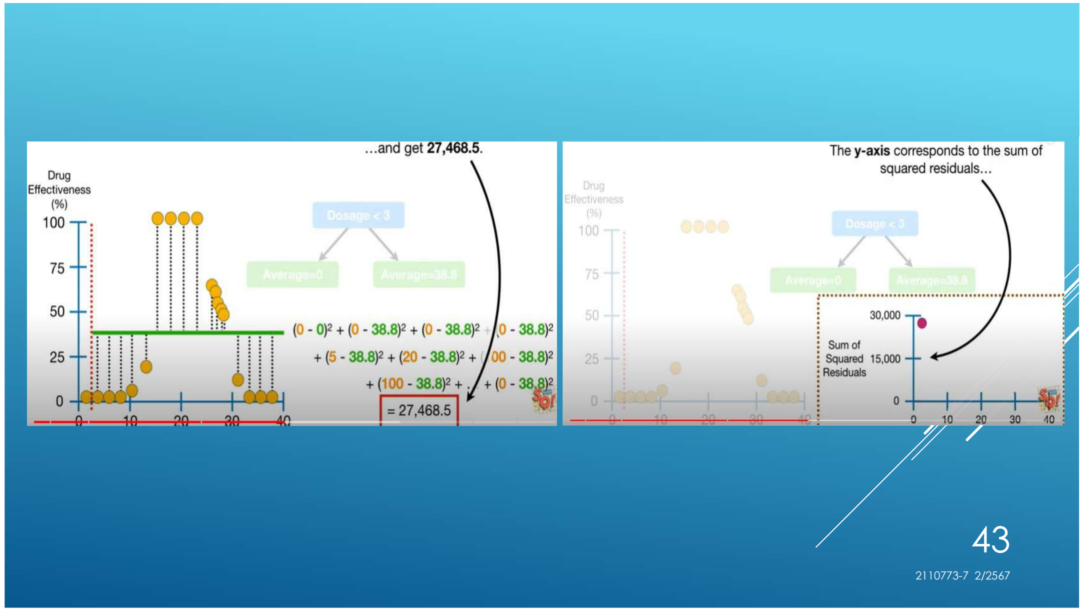

…moderate dosages work really well…
…somewhat higher dosages work at about **50%** effectiveness…
…and high dosages are not effective at all.
Drug Effectiveness (%)
38
2110773-7 2/2567

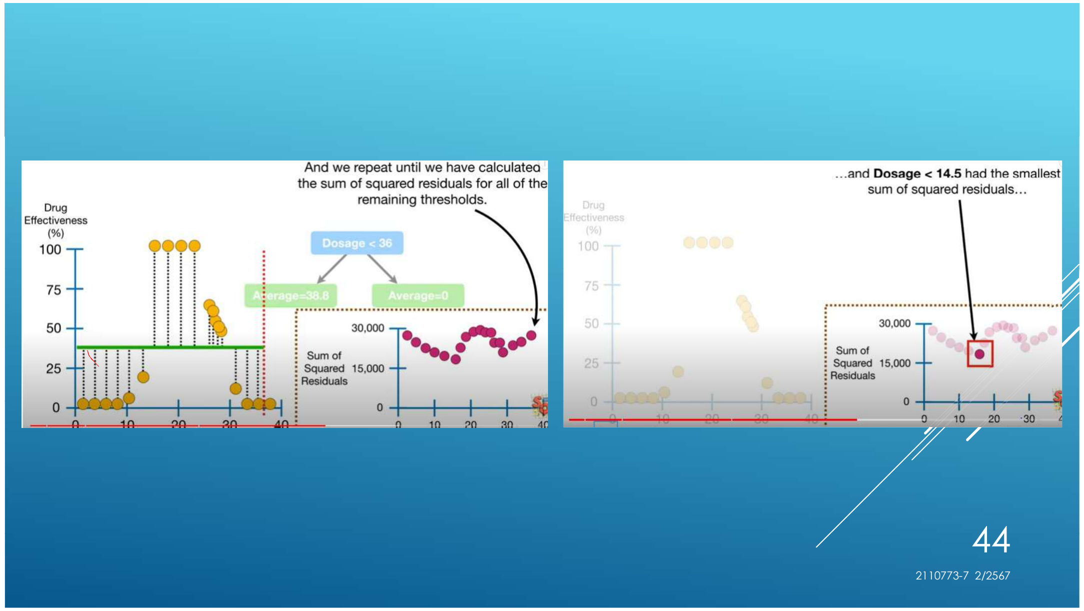

For example, if someone told us they were taking a 20 mg Dose…
…even though the observed data says that it should be **100%** **Effective**.
Drug Effectiveness (%)
One option is to use a **Regression Tree**.
Dosage < 14.5
4.2% Effective
Dosage >= 29
2.5% Effective
Dosage >= 23.5
52.8% Effective
39
2110773-7 2/2567

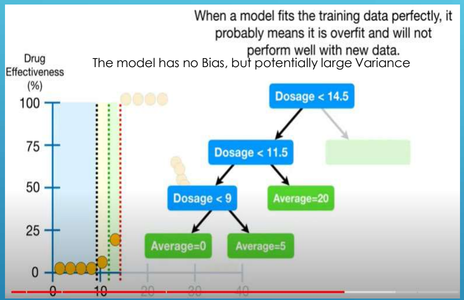

…so the tree uses the average value, **4.2%**, as its prediction for people with **Dosages < 14.5**.
…so the tree uses the average value, **2.5%**, as its prediction for people with **Dosages >= 29**.
Drug Effectiveness (%)
Dosage < 14.5
4.2% Effective
Dosage >= 29
2.5% Effective
40
2110773-7 2/2567

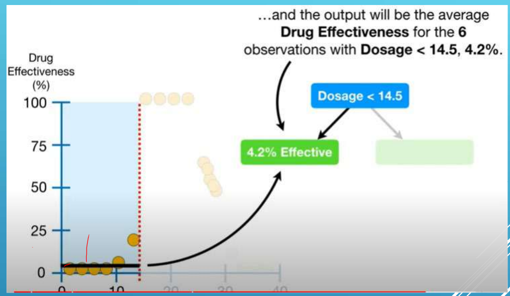

...so the tree uses the average value, 52.8%, as its prediction for people with Dosages between 23.5 and 29.
Drug Effectiveness (%)
Dosage < 14.5
Dosage >= 29
Dosage >= 23.5
52.8% Effective
...so the tree uses the average value, 100%, as its prediction for people with Dosages between 14.5 and 23.5.
100% Effective
41
2110773-7 2/2567


...so we put 38.8 in the leaf on the right side, for when the Dosage >= 3.
Drug Effectiveness (%)
Dosage < 3
Average=0
Average=38.8
In other words, this dotted line is a residual.
42
2110773-7 2/2567


...and get 27,468.5.
Drug Effectiveness (%)
Dosage < 3
Average=0
Average=38.8
= 27,468.5
The y-axis corresponds to the sum of squared residuals...
Sum of Squared Residuals
43
2110773-7 2/2567

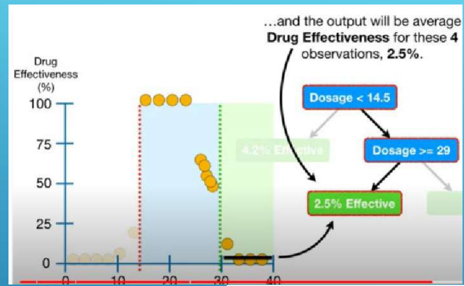

And we repeat until we have calculated the sum of squared residuals for all of the remaining thresholds.
Drug Effectiveness (%)
Dosage < 36
Average=38.8
Average=0
Sum of Squared Residuals
...and Dosage < 14.5 had the smallest sum of squared residuals...
44
2110773-7 2/2567

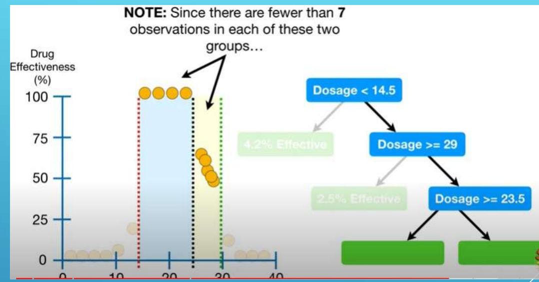


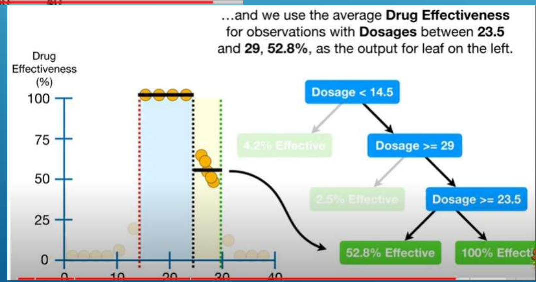


The simplest is to only split observations when there are more than some minimum number. Typically, the minimum number of observations to allow for a split is **20**.

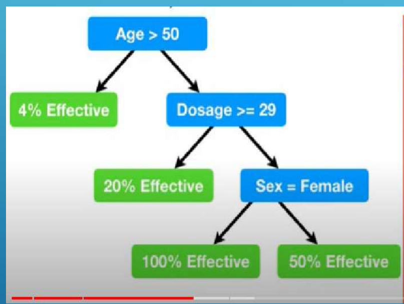
However, since this example doesn't have many observations, I set the minimum to **7**.







- 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

Pro

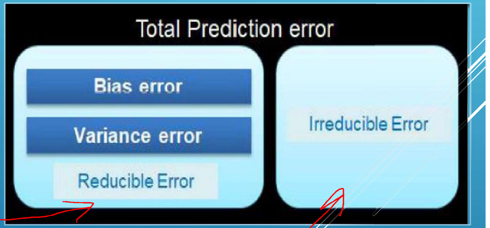
- easy to interpret and visualize
- easily capture Non-linear patterns with non-parametric nature of the algorithm.
- requires fewer data preprocessing, no need to normalize features
- **can be applied for variable selection**

Con

- Sensitive to noisy data. It can overfit noisy data.
- Biased with imbalanced dataset, balance out the dataset before creating DT is recommended
- **small variation(or variance) in data can result in different DT. This can be reduced by bagging and boosting algorithms.**

DT CLASSIFIER

- Every model has both bias and variance error components in addition to white noise.
- The ideal model will have both low bias and low variance.
- Unfortunately, bias and variance are inversely related to each other; while trying to reduce one component, the other component of the model will increase.
- The true art lies in creating a good fit by balancing both.



$$E\left(y_0 - \hat{f}(x_0)\right)^2 = Var\left(\hat{f}(x_0)\right) + \left[Bias(\hat{f}(x_0))\right]^2 + Var(\varepsilon)$$

Bias-error

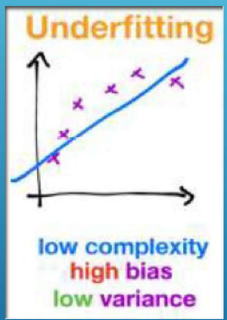
- Difference between predicted and actual data points caused by **oversimplified** model or unable to capture underlying pattern of data.
- It misses how the features in the training data set relate to the expected output.
- A model with high bias is too simple and has low number of predictors.

Variance-error

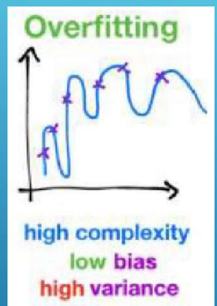
- High variance error of a model implies that it is highly sensitive to small fluctuations. This model flounders outside of its comfort zone(training data)
- Any model which has very large number of predictors will end up being a very **complex model** which will deliver very accurate predictions for the training data that it has seen already but this complexity makes the generalization of this model to unseen data very difficult, i.e. a high variance model. Thus, this model will perform very poorly on test data.

REDUCIBLE ERROR/ INADVERTENT MISTAKES

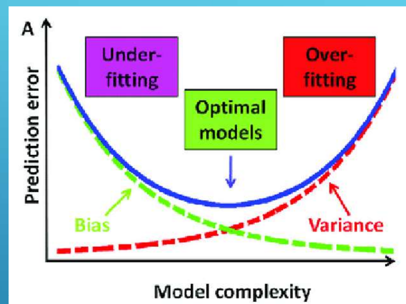
Bias-error



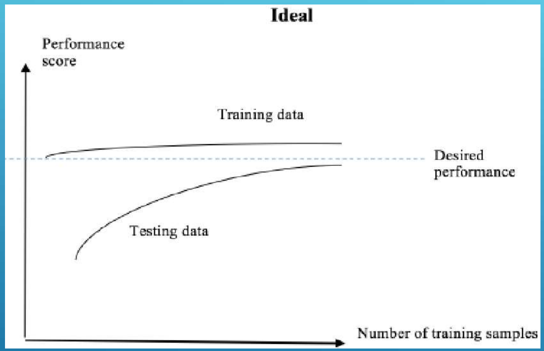
Variance-error



- On the one hand, we want our algorithm to model the training data very closely, otherwise we'll miss relevant features and interesting trends.
- However, on the other hand we don't want our model to fit too closely, and risk overinterpreting every outlier and irregularity.
- **High-Bias**: Suggests more assumptions about the form of the target function.
- **High Variance**: Suggests large changes to the estimate of the target function with changes to the training dataset.



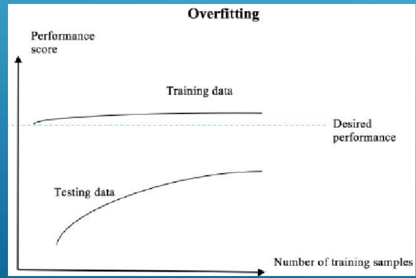
BIAS-VARIANCE DILEMMA



- A **learning curve** is usually used to evaluate the bias and variance of a model.
- For a model that fits well on the training samples, the performance of training samples should be above desire. Ideally, as the number of training samples increases, the model performance on testing samples improves; eventually the performance on testing samples becomes close to that on training samples.

DIAGNOSING OVERFITTING AND UNDERFITTING

When the performance on testing samples converges at a value far from the performance on training samples, overfitting can be concluded. In this case, the model fails to generalize to instances that are not seen.



For a model that does not even fit well on the training samples, underfitting is easily spotted: both performances on training and testing samples are below desire in the learning curve.