Decision Tree

Hossam Ahmed Salah

MSP *Helwan*

Agenda

Variance and Bias

Overfitting and Underfitting

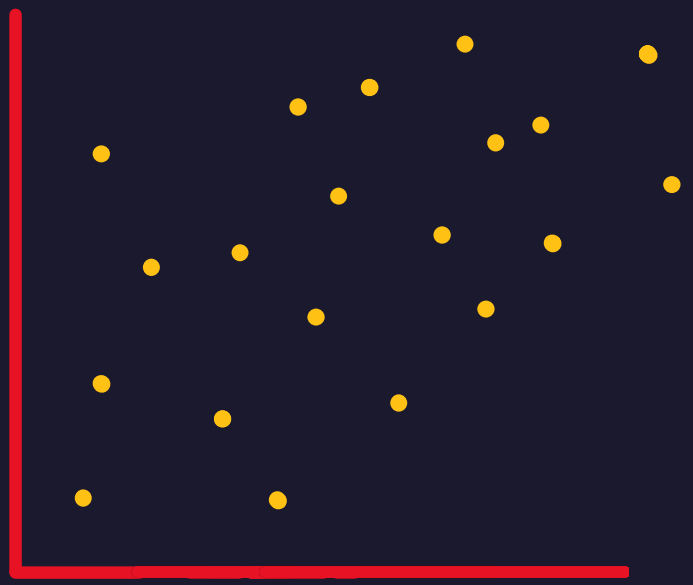
Classification Decision Tree

Regression Decision Tree

Pruning a tree

Variance

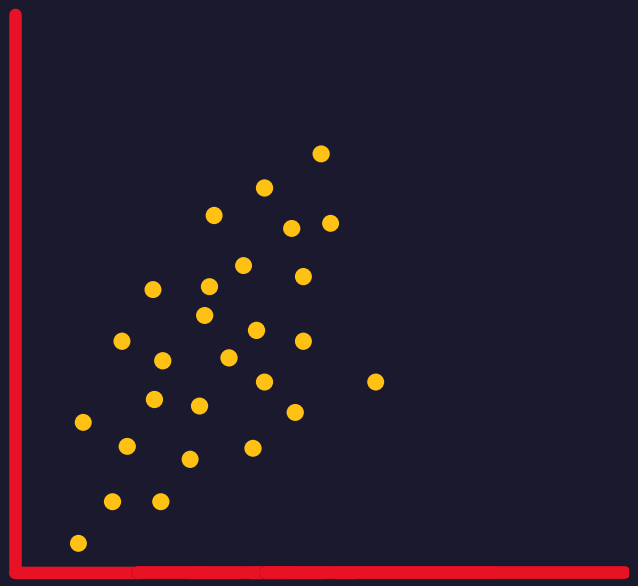
- Measurement of spread in the dataset
- مقياس لتباعد النقاط المختلفة من النقطة الوسط



High Variance

Variance

- Measurement of spread in the dataset
- مقياس لتباعد النقاط المختلفة من النقطة الوسط



Low Variance

Bias Error

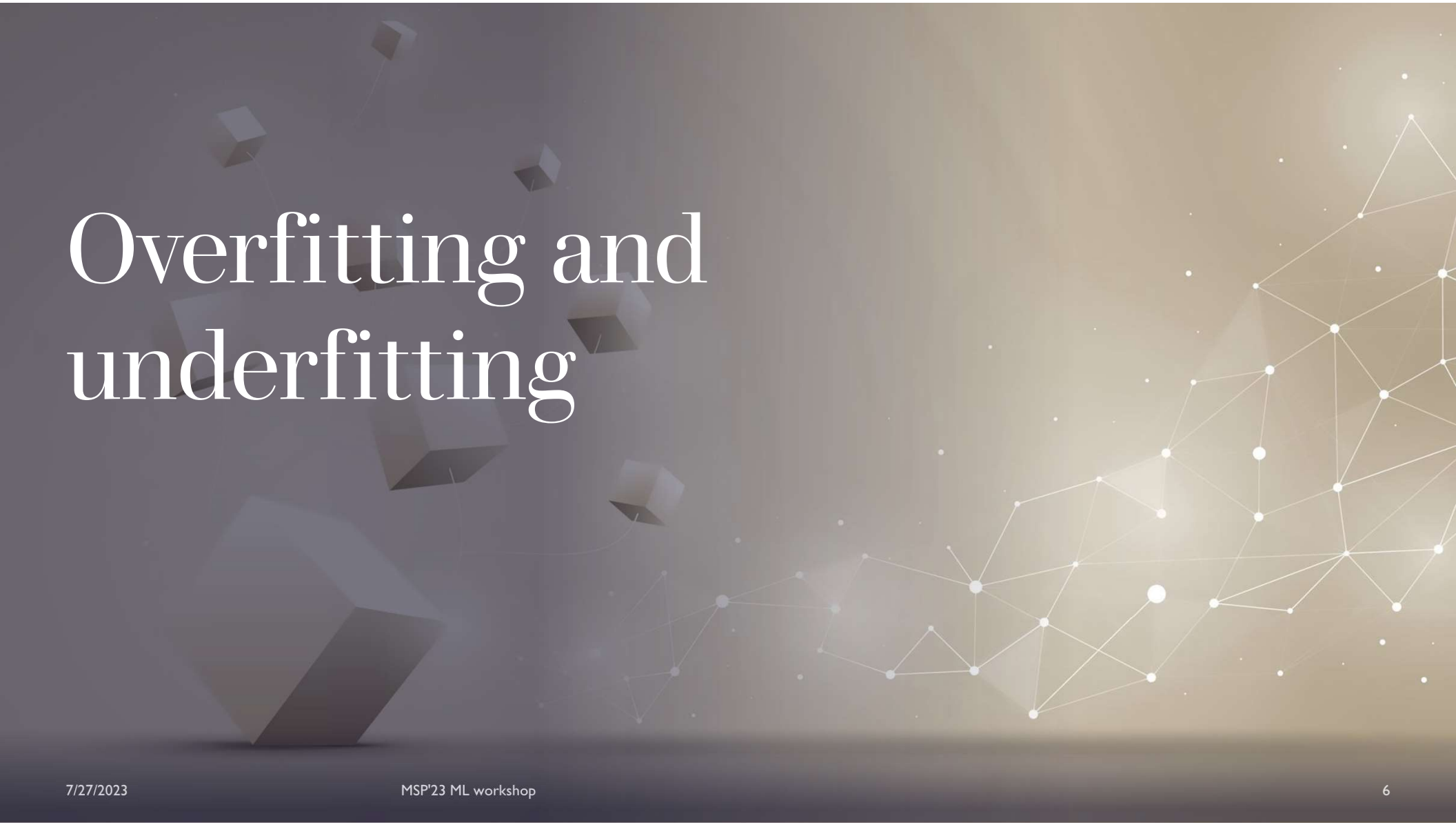
In statistics, bias refers to the tendency of a statistical estimator or method to consistently overestimate or underestimate a population parameter

tendency which causes differences between results and facts

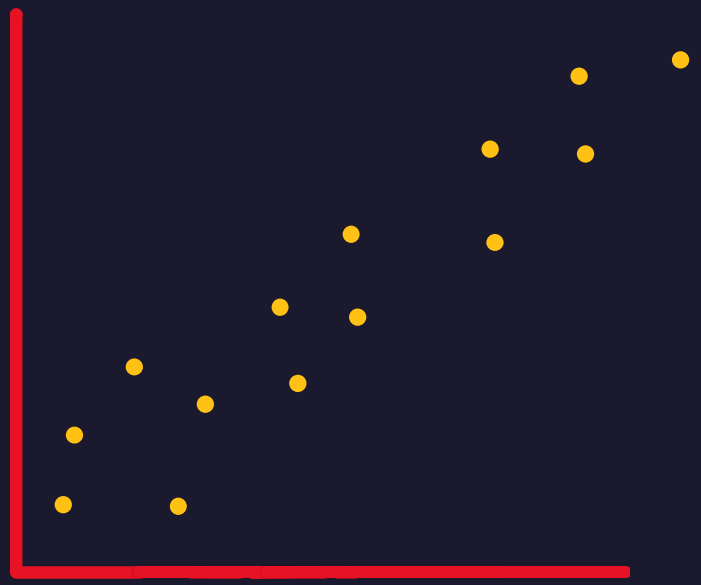
Bias in ML is a sort of mistake in which some aspects of a dataset are given more weight and/or representation than others

In ML bias inability to capture the underlying complexity of the data.

Overfitting and underfitting



- Considering we have this data, and we want to train a 2 models



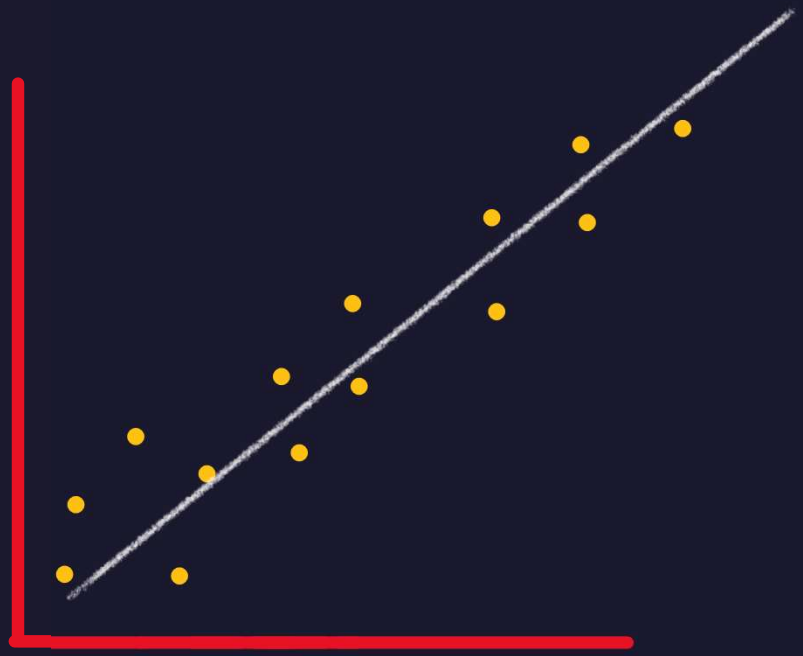
Models

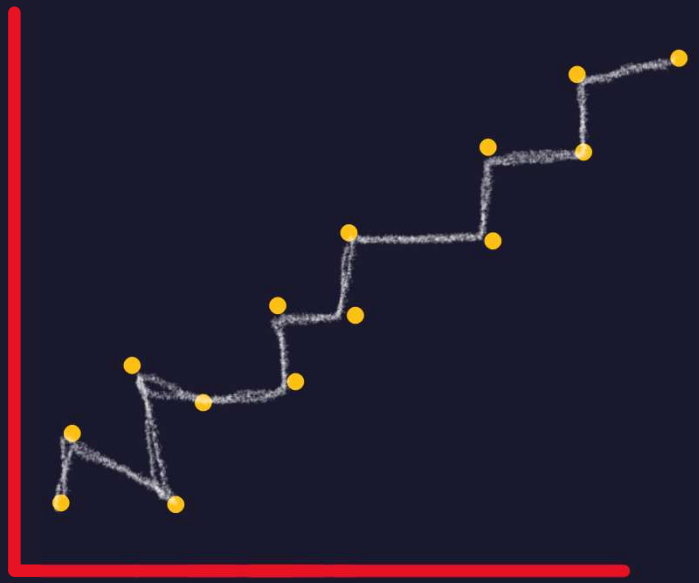
- Linear
- Polynomial

$$y = \beta_0 + \beta_1 x$$

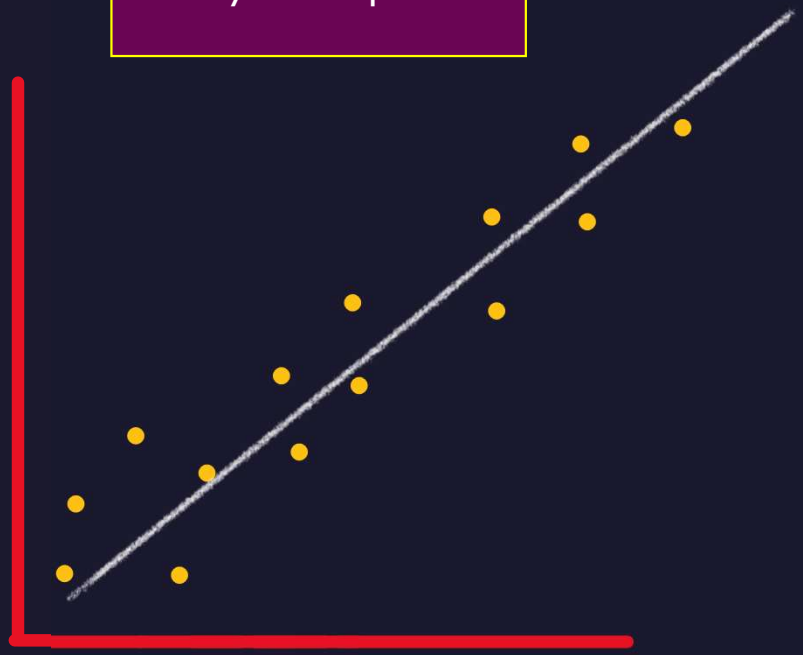
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n$$

- Considering we have this data, and we want to train a 2 models

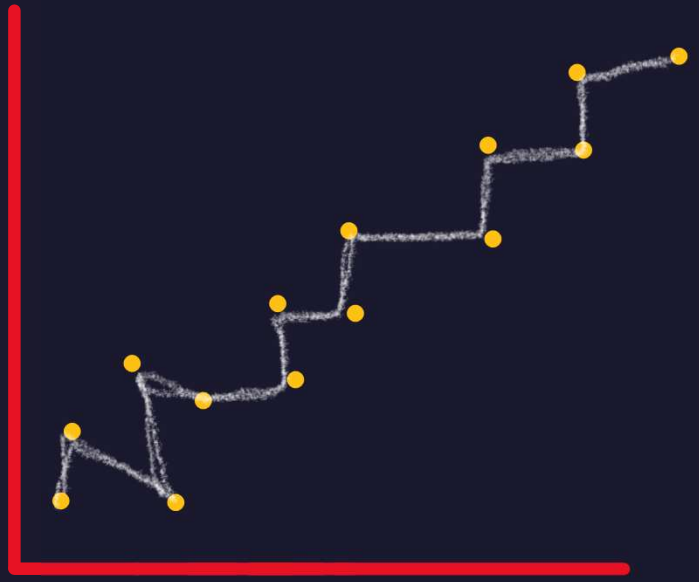




Only fit few points

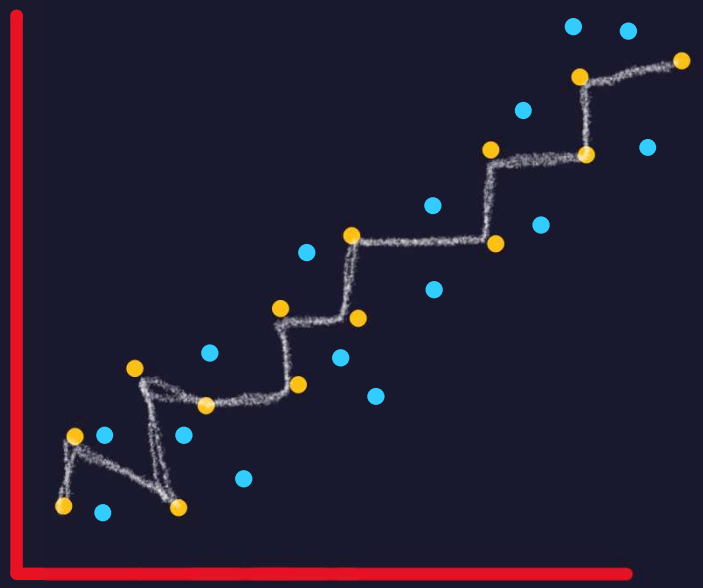


Perfect fit of the trained data 100%



- So, you decided now to deploy your perfect model on the real world

- So, a lot of data inputted to your model

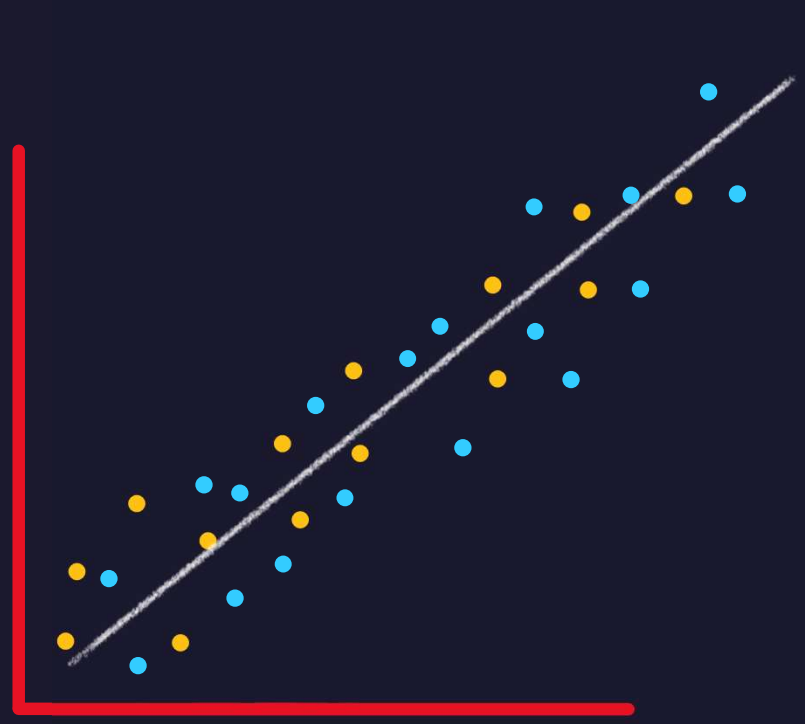


Your model is terrible on real data not near even

Seeking perfection in train set lead to **high variance** model

This called **overfitting**

- What about the linear Model



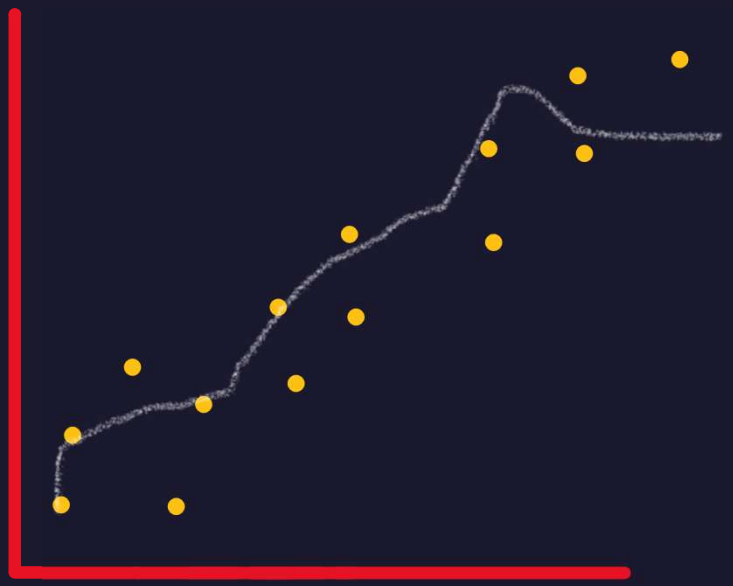
Terrible model but we the training error and testing error is not too far

If average error is 20% for each point you can say that new points can be + or – 20%

This model has a **high bias** following the line doesn't care about the data

This model **underfit** the data too simple to model it

- We need a mode that generalize
  - we shall tolerant some bias (error) in the training set
  - That mean making this model has less variance
  - It's to simplify the model

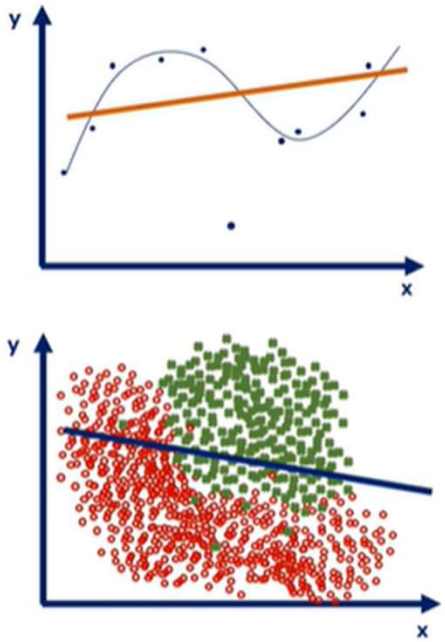


We have errors in the training set
But we can have less error when dealing with new input

A Best fit trying to achieve balance between Bias(error, simplicity) and Variance (complexity)

It's a tradeoff

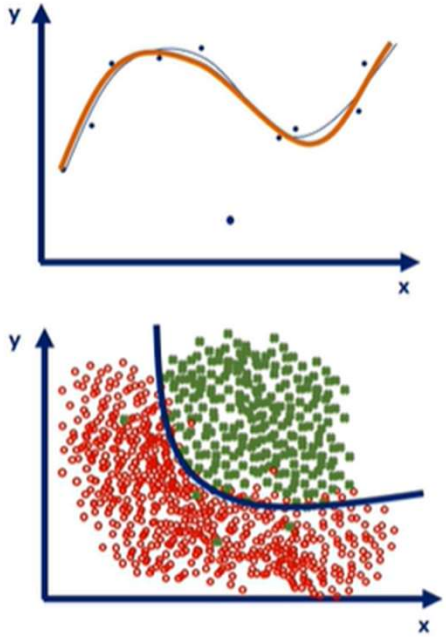
An **underfitted** model



Doesn't capture any logic

- High loss
- Low accuracy

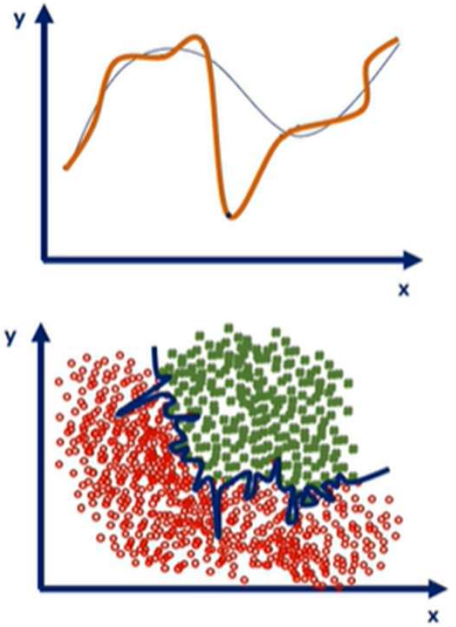
A **good** model



Captures the underlying logic of the dataset

- Low loss
- High accuracy

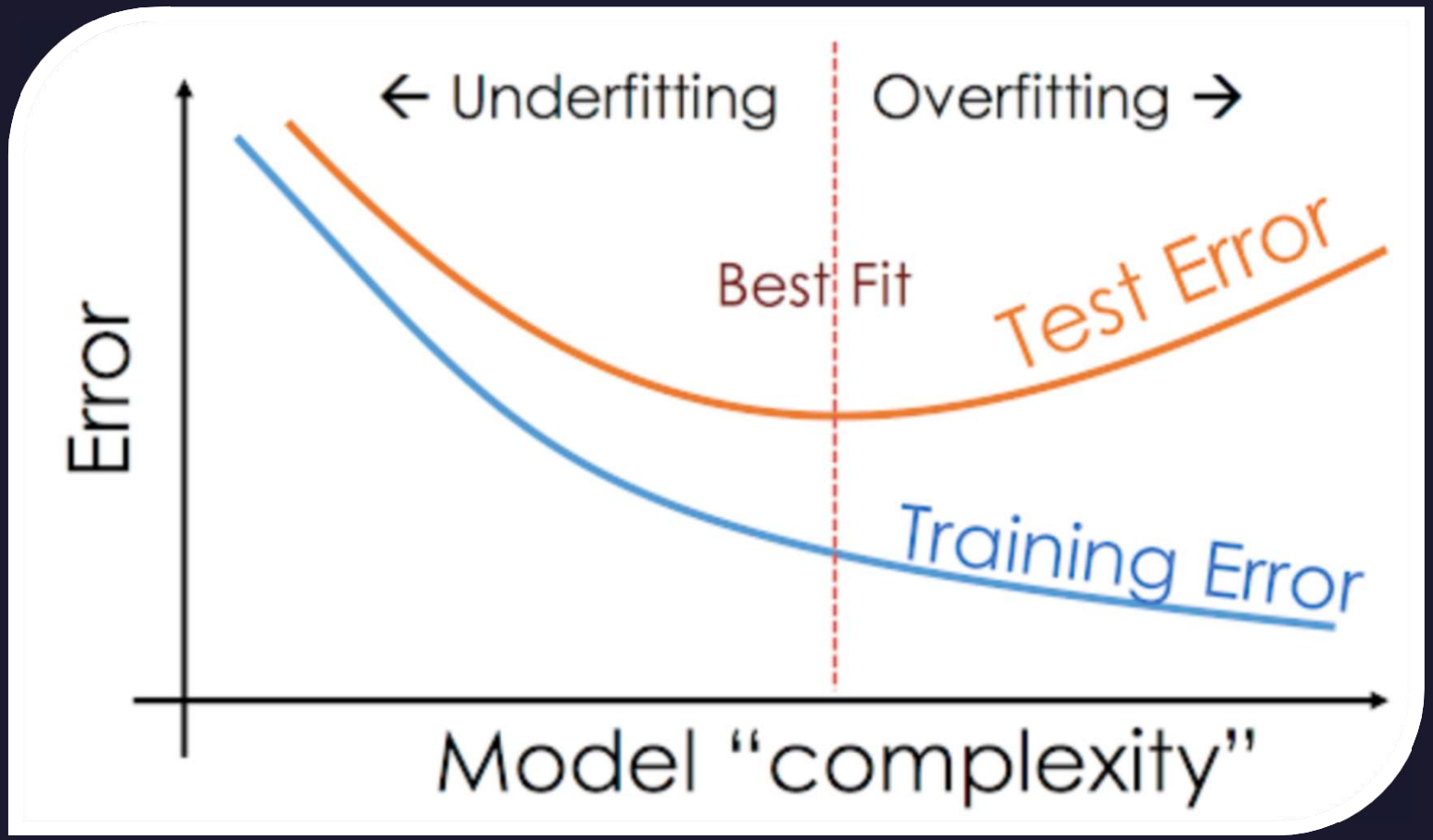
An **overfitted** model



Captures all the noise, thus "missed the point"

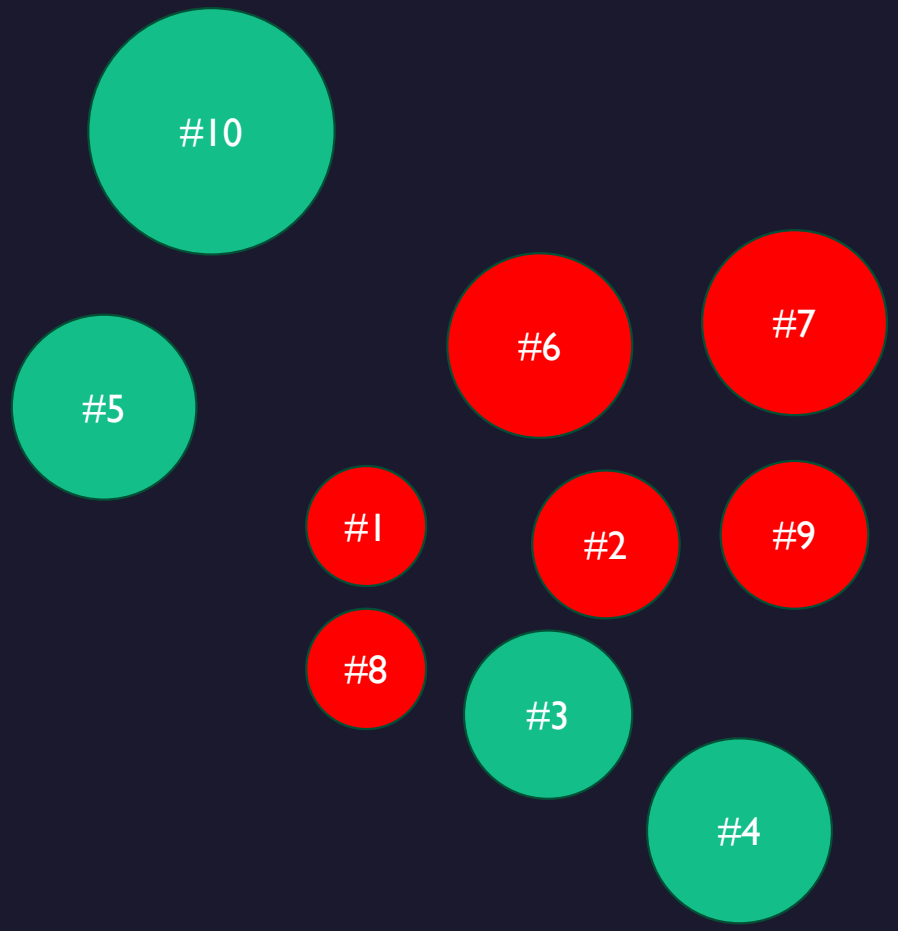
- Low loss
- Low accuracy

Bias-variance tradeoff: The balance between underfitting and overfitting


← Underfitting
Overfitting →
Error
Best Fit
Test Error
Training Error
Model "complexity"

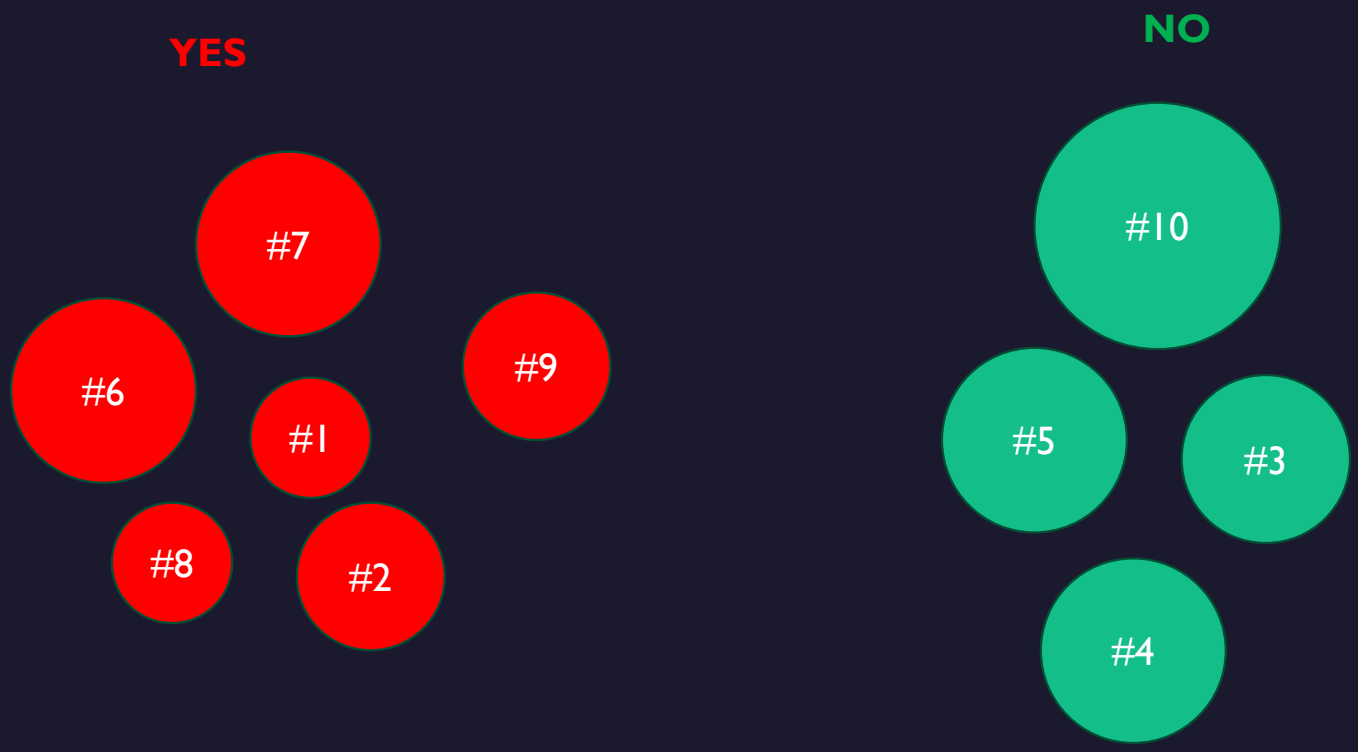
Classification Decision Tree

Circle#	color	r
1	R	1
2	R	3
3	G	4
4	G	5
5	G	6
6	R	7
7	R	7
8	R	1
9	R	3
10	G	8



Is you color RED?

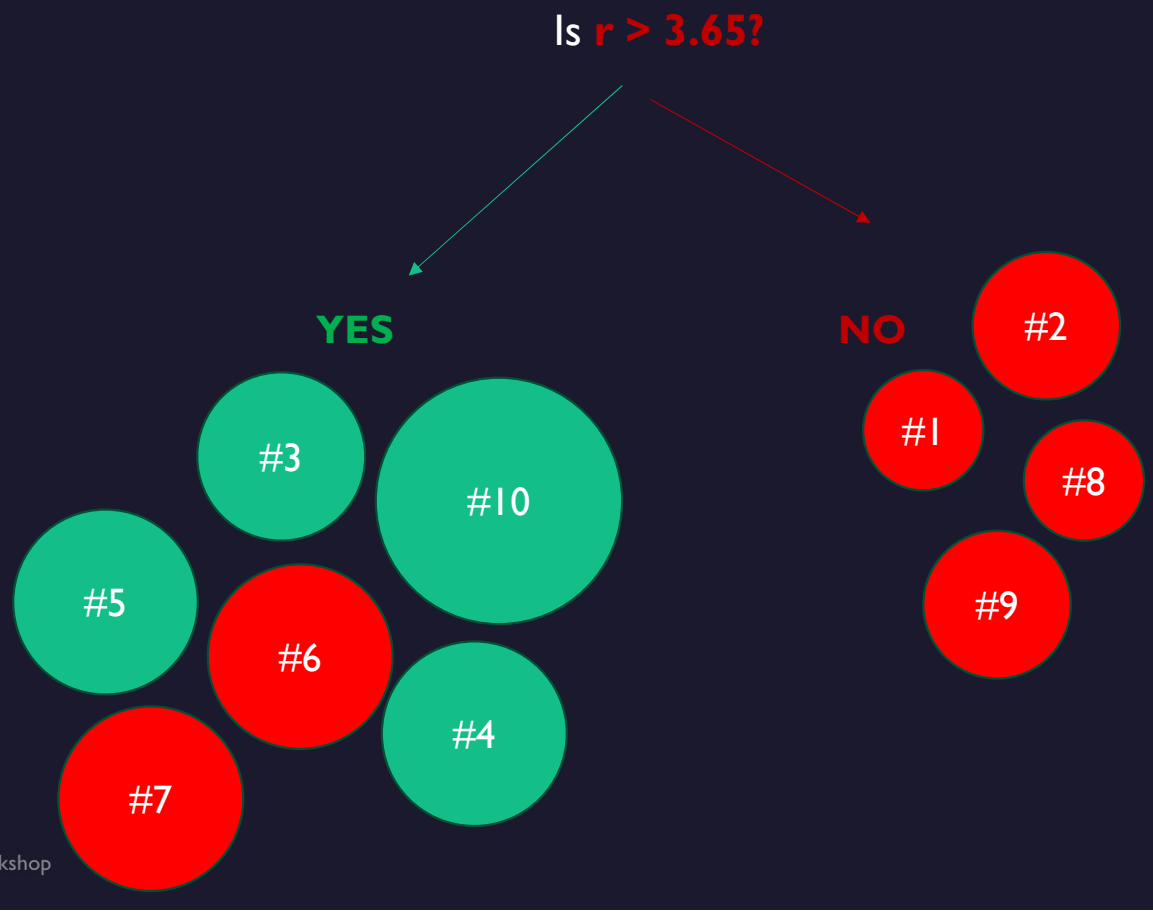
We were a little bit lucky ? Choosing color was the best to separate the circles

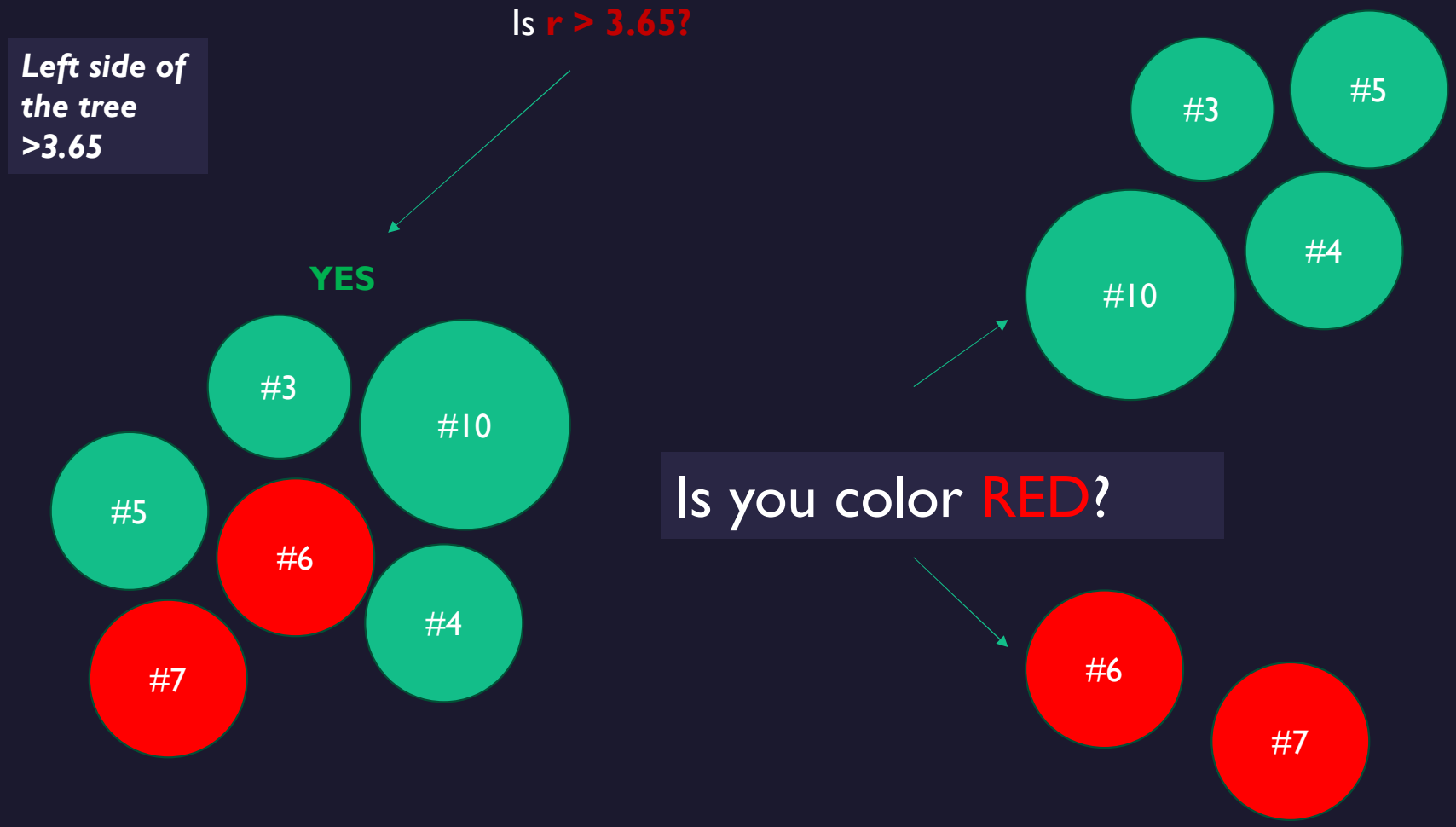


Circle#	color	r
1	R	1
2	R	3
3	G	4
4	G	5
5	G	6
6	R	7
7	R	7
8	R	1
9	R	3
10	G	8

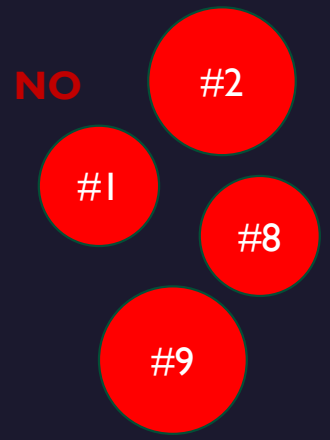
What if we tried to split with r based on threshold **3.65**

Circle#	color	r
1	R	1
2	R	3
3	G	4
4	G	5
5	G	6
6	R	7
7	R	7
8	R	1
9	R	3
10	G	8

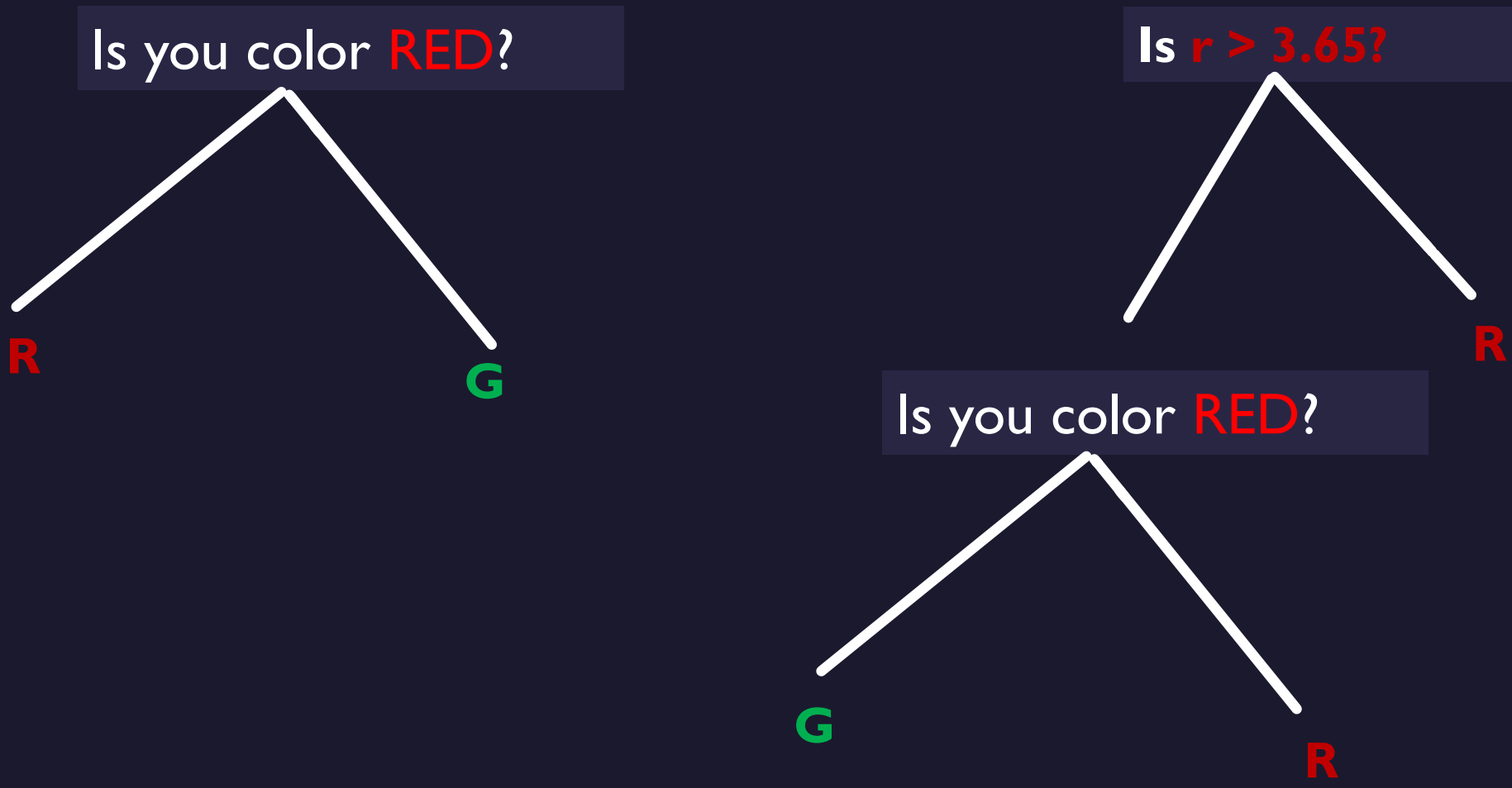



Is r > 3.65?
Left side of the tree >3.65
YES
#3
#10
#5
#6
#4
#7
#3
#5
#10
#4
Is you color RED?
#6
#7

Is **r > 3.65?**



right side of the tree <=3.65


Is you color RED?
R
G
Is r > 3.65?
R
Is you color RED?
G
R

From this simple example we see that the splitting column can affect the tree size , so selecting the best feature to split is the challenge in DTs

Also, we can notice that the best feature to split is the one that give us more **pure nodes**

We can see that here simply because we have 2 colors only but what if we got three colors or more classes in each feature

Entropy and **Gini index** two ways decide how to split , by selecting the split give us purest nodes as possible

DT features and notes

- Decision trees use a **top-down** approach called recursive binary splitting.
- Recursive binary splitting starts at the top of the tree and splits the predictor space into two branches.
- The algorithm is greedy because it selects the best split at each step, without considering future splits.
- The algorithm evaluates variables based on statistical criteria to choose the variable that performs best. {Entropy , Gini index}
- The predictor space is divided into two branches at each split, creating a hierarchical structure.
- The algorithm does not project forward to optimize the entire tree, but rather focuses on the current step.

Gini index

- The **Gini Index measures the probability of misclassification** for a randomly chosen instance.
- **A lower Gini Index indicates a better split with a lower likelihood of misclassification.**
- The **Gini Index approach** focuses on **measuring impurity**

$$Gini = 1 - \sum_{i=i}^{n} P_i$$

- The Gini Index ranges **from 0 (highest purity) to 0.5 (random assignment of classes).**
- To calculate the Gini Index for a node, the Gini Index is **calculated for each sub node**, and then a weighted average is taken to determine the overall Gini Index for the node

Gini index

the root node : Student Background

#	Result	Other online courses	Background	Working
1	Pass	y	Math	NW
2	Fail	n	Math	W
3	Fail	y	Math	W
4	Pass	y	Cs	NW
5	Fail	n	Other	W
6	Fail	y	Other	W
7	Pass	y	Math	NW
8	Pass	y	Cs	NW
9	Pass	n	Math	W
10	Pass	n	Cs	W
11	Pass	y	Cs	W
12	Pass	n	Math	NW
13	Fail	y	Other	W
14	Fail	n	Other	NW
15	Fail	n	Math	W

1. **Calculate Gini index for each sub node**
   - **Math (total observation 7)**
     - **4 pass, 3 fail**
     - **1 -**$(P(pass|\ Math))^2-(P(fail|Math))^2$
     - **1 –** $\left(\frac{4}{7}\right)^2$ **-**$\left(\frac{3}{7}\right)^2$ = 0.48979
   - **CS (total observation 4)**
     - **4 pass, 0 fail**
     - **1 -**$(P(pass|\ Cs))^2-(P(fail|Cs))^2$
     - **1 –** $\left(\frac{4}{4}\right)^2$ **-**$\left(\frac{0}{4}\right)^2$ = 0
   - **Other (total observation 4)**
     - **0 pass, 1 fail**
     - **1 -**$(P(pass|\ Other))^2-(P(fail|Other))^2$
     - **1 –** $\left(\frac{0}{4}\right)^2$ **-**$\left(\frac{4}{4}\right)^2$ = 0
2. **Gini**$_{background}$ **=** $\frac{7}{15}$**×0.48979+**$\frac{4}{15}$**×0+**$\frac{4}{15}$**×0 =** 0.22857

Gini index

the root node :Working status

#	Result	Other online courses	Background	Working
1	Pass	y	Math	NW
2	Fail	n	Math	W
3	Fail	y	Math	W
4	Pass	y	Cs	NW
5	Fail	n	Other	W
6	Fail	y	Other	W
7	Pass	y	Math	NW
8	Pass	y	Cs	NW
9	Pass	n	Math	W
10	Pass	n	Cs	W
11	Pass	y	Cs	W
12	Pass	n	Math	NW
13	Fail	y	Other	W
14	Fail	n	Other	NW
15	Fail	n	Math	W

1. **Calculate Gini index for each sub node**
   - **Working (total observation 9)**
     - **6 pass, 3 fail**
     - **1 -**$(P(pass|Work))^2-(P(fail|Work))^2$
     - **1 –** $\left(\frac{6}{9}\right)^2$ **-**$\left(\frac{3}{9}\right)^2$ = 0.44444
   - **Not Working (total observation 5)**
     - **4 pass, 1 fail**
     - **1 -**$(P(pass|NW))^2-(P(fail|NW))^2$
     - **1 –** $\left(\frac{4}{6}\right)^2$ **-**$\left(\frac{1}{6}\right)^2$ = 0.52778
2. **Gini**$_{WorkingStatus}$ **=** $\frac{9}{15}$**×**0.44444+$\frac{5}{15}$×0.52778= 0.44259

Gini index

the root node :Other online courses

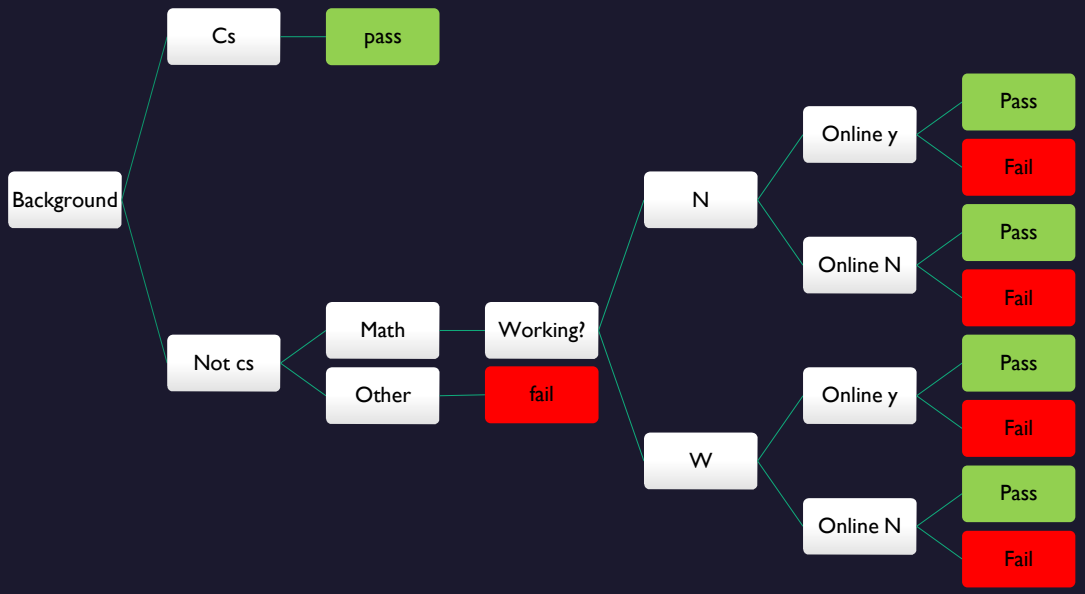
#	Result	Other online courses	Background	Working
1	Pass	y	Math	NW
2	Fail	n	Math	W
3	Fail	y	Math	W
4	Pass	y	Cs	NW
5	Fail	n	Other	W
6	Fail	y	Other	W
7	Pass	y	Math	NW
8	Pass	y	Cs	NW
9	Pass	n	Math	W
10	Pass	n	Cs	W
11	Pass	y	Cs	W
12	Pass	n	Math	NW
13	Fail	y	Other	W
14	Fail	n	Other	NW
15	Fail	n	Math	W

1. **Calculate Gini index for each sub node**
   - **Yes (total observation 8)**
     - **5 pass, 3 fail**
     - **1 -**$(P(pass|\ Yes))^2-(P(fail|Yes))^2$
     - **1 –** $\left(\frac{5}{8}\right)^2$ **-**$\left(\frac{3}{8}\right)^2$ = 0.46875
   - **No (total observation 7)**
     - **3 pass, 4 fail**
     - **1 -**$(P(pass|\ No))^2-(P(fail|No))^2$
     - **1 –** $\left(\frac{3}{7}\right)^2$ **-**$\left(\frac{5}{7}\right)^2$ = 0.48798
2. **Gini**$_{OnlineCourses}$ **=** $\frac{8}{15}$**×**0.46875+$\frac{7}{15}$×0.48798= 0.47825

Gini index

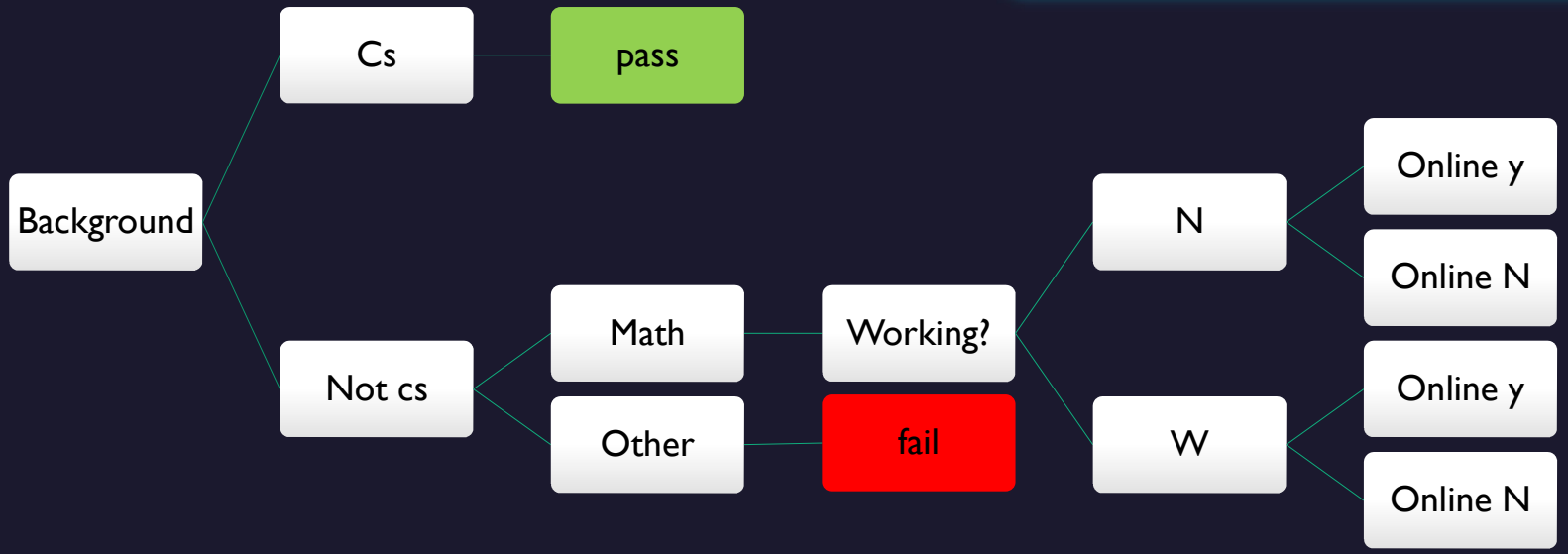
#	Result	Other online courses	Background	Working
1	Pass	y	Math	NW
2	Fail	n	Math	W
3	Fail	y	Math	W
4	Pass	y	Cs	NW
5	Fail	n	Other	W
6	Fail	y	Other	W
7	Pass	y	Math	NW
8	Pass	y	Cs	NW
9	Pass	n	Math	W
10	Pass	n	Cs	W
11	Pass	y	Cs	W
12	Pass	n	Math	NW
13	Fail	y	Other	W
14	Fail	n	Other	NW
15	Fail	n	Math	W

1. $\text{Gini}_{OnlineCourses} = \frac{8}{15}\times 0.46875 + \frac{7}{15}\times 0.48798 = 0.47825$
2. $\text{Gini}_{WorkingStatus} = \frac{9}{15}\times 0.44444 + \frac{5}{15}\times 0.52778 = 0.44259$
3. $\text{Gini}_{background} = \frac{7}{15}\times 0.48979 + \frac{4}{15}\times 0 + \frac{4}{15}\times 0 = 0.22857$



Gini index

We may not get all leaves pure single class pass/fail , maybe tree terminated with heterogeneous classes here we may classify based on the majority class in the leaf



Gini index

A full decision tree is a tree that is grown until all the terminal nodes (i.e., the leaves) contain a **minimum number of observations**, or until all the observations in the training set belong to **the same class**. However, even if a decision tree is grown to its maximum depth, there may still be some leaves that are not pure if the algorithm determines that *further splitting of the data would not significantly improve the classification accuracy*

Entropy

- ***In information theory, the entropy of a random variable is the average level of "information", "surprise", or "uncertainty" inherent to the variable's possible outcome***
- It is often associated with a state of disorder, randomness, or uncertainty.
- Entropy in Decision Trees is a measure of disorder or impurity in a node.
- **Nodes with a *more diverse composition have higher entropy* than nodes with a single category.**
- Entropy ranges from 0 to 1, with **0 representing minimum entropy (pure node)** and **1 representing maximum entropy (high disorder).**
- Entropy helps determine the homogeneity or purity of a node in Decision Trees.

Entropy

$$E = -\sum_{i=1}^{n} P_i \log_2(P_i)$$

- **Information Gain** measures the amount of information a feature provides for predicting the target variable.

$$Information\ Gain = Entropy_{\ parent} - Entropy_{\ children}$$

Entropy

$P(pass) = 8/15 = 0.5333$
$P(fail) = 7/15 = 0.4666 P(fail|student\ background) = 7/15 = 0.4666$

1. calculate the Entropy

$$E = -\sum_{i=1}^{2} P_i \times \log_2 P_i =$$
$$-[0.5333 \times \log_2 0.533 + 0.4666 \times \log_2 0.4666] = 0.99679$$

Entropy of the column parent
P(target = T/F)

Entropy for the column classes (childrens)
P(target|column class)
class := eg. true/false...

$$E_{working} = -[\frac{3}{9} \times \log_2(3/9) + \frac{6}{9} \times \log_2(6/9)] = 0.918295$$

$$E_{Not\ Working} = -[5/6 \times \log_2(5/6) + 1/6 \times \log_2(1/6)] = 0.65002$$

you have 8 pass, 7 fail

you have 9 students Work and 6 not working

$P(pass|working) = [P(pass) \cap P(work)]/P(work) = 3/9$

كام واحد شغال ونجح على عدد ال شغالين كلهم

$P(fial|working) = 6/9$

$P(pass|Not\ working) = 5/6$
$P(fail|Not\ working) = 1/6$

Average Entropy of the classes of the column

$$E_{working-status} = [\frac{working}{15} \times Entropy_{working} + \frac{not\ working}{15} \times Entropy_{Not\ working}]$$

$$E_{working-status} = [\frac{9}{15} \times 0.918295 + \frac{6}{15} \times 0.65002] = 0.8109$$

calculate information gain

$InformationGain = Entropy_{Parent} — Entropy_{child} = 0.99679–0.8109 = 0.18589$

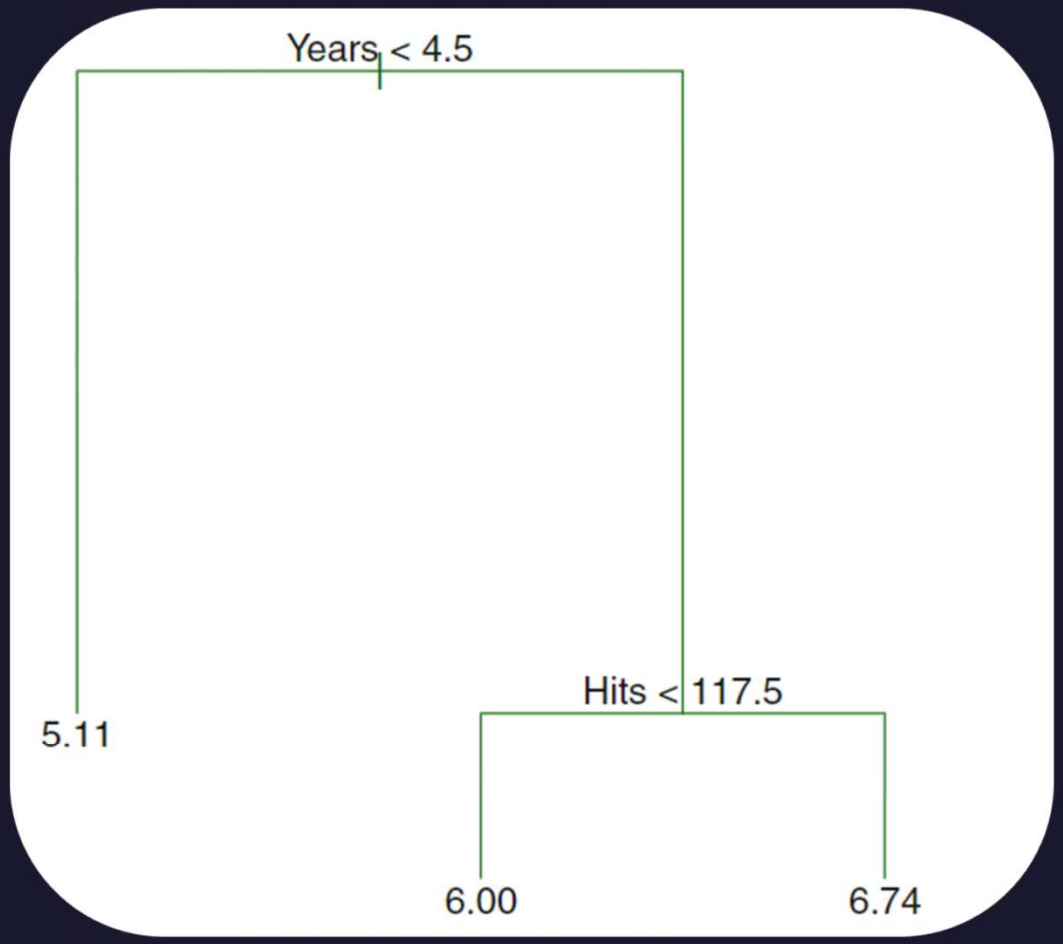
$$Information\ Gain = E_{parent} - AvgE_{child}$$

Do for each column and select to split with higher info gain

	Entropy Node	Average Entropy	Information Gain
Parent	0.9968		
working	0.9183	0.8110	
Not_work	0.6500		0.1858
Bkgrd_Ma	0.9852		
Bkgrd_CS	0.0000	0.4598	
Bkgrd_oth	0.0000		0.5370
online_co	0.9544	0.9688	
online_no	0.9852		0.0280

Regression Decision tree

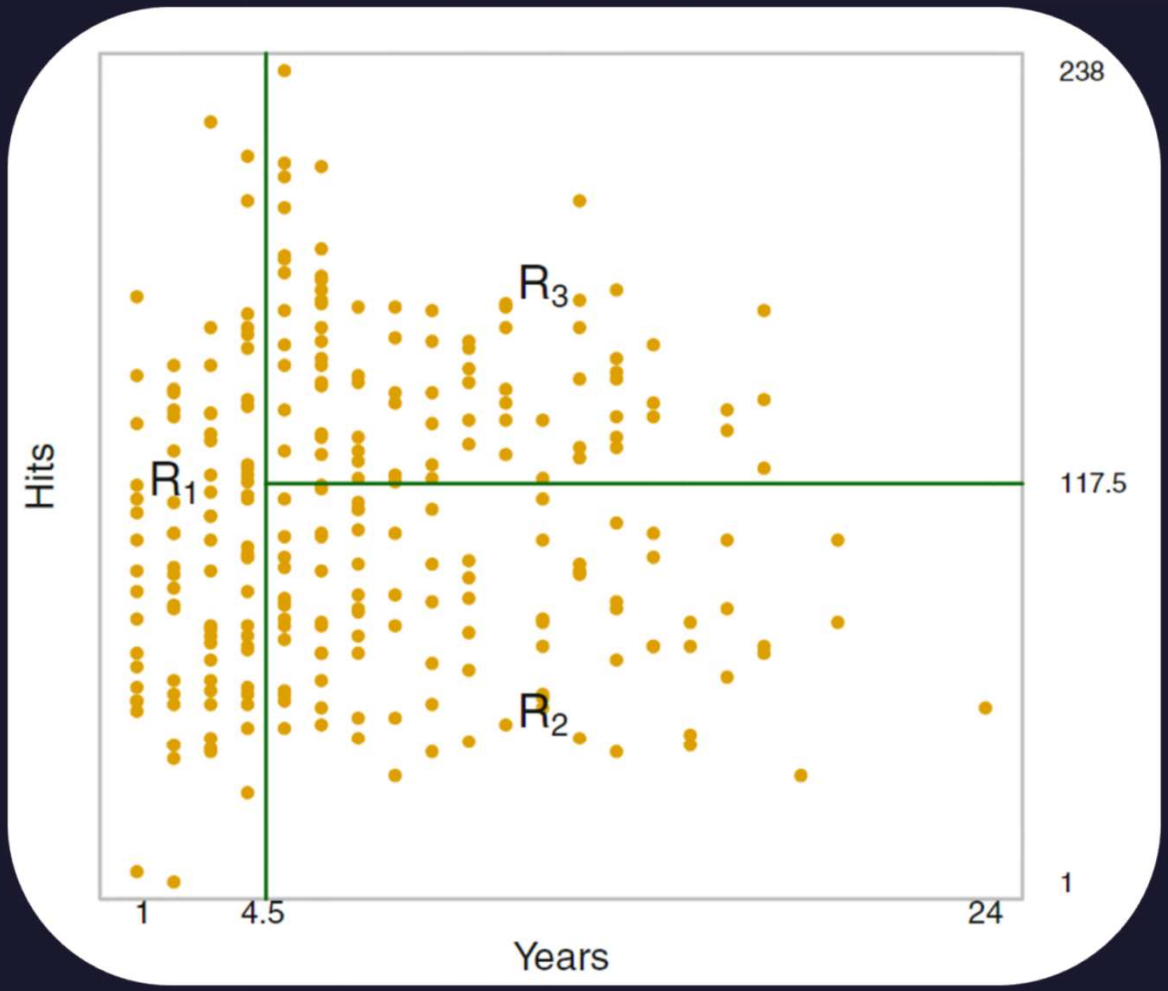
Baseball



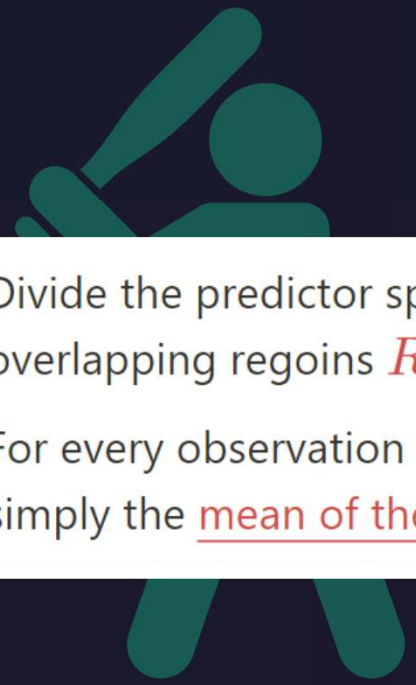
Baseball

- The regression tree predicts the log salary of baseball players based on two predictor variables: **years played, and hits** made the previous year.
- The first split is on the Years variable, splitting players into those with less than 4.5 years played versus 4.5 or more years played.
- The second split is on the Hits variable, further splitting the two branches based on a threshold on hits made.
- There are three terminal nodes or leaves, indicating the final predicted values for log salary. The number in each leaf node is the mean log salary for that group of players.
- The tree performs splits by choosing the predictor and threshold that results in the most homogeneous groups in terms of the response variable (log salary)

Baseball



- We now discuss the process of building a regression tree. Roughly speaking, th**ere are two steps**



1. Divide the predictor space , set of possible values for $X_1, X_2, \ldots, X_p$ into J distincit and non overlapping regoins R_1, R_2, R_3
2. For every observation that falls into the region R_j , we make the same prediction, which is simply the mean of the response values for the training observations in R_j

Baseball

- In the example, there are two regions R1 and R2. Observations in R1 get a prediction of 10, while those in R2 get a prediction of 20.
- The regions can have any shape in theory, but for simplicity and interpretability, the predictor space is divided into rectangular regions.
- The goal is to find rectangular regions that minimize the residual sum of squares (RSS), which measures how close the predictions are to the actual responses.
- By minimizing the RSS, the tree finds the partition of the predictor space that results in the most homogeneous regions in terms of the response variable.
- At each split or node in the tree, the predictors and split points are chosen to maximize the decrease in RSS (or increase in homogeneity).

Baseball

- RSS
- Residual(e) : ***difference*** between an **observed value** and the **predicted value** of a variable.
- `RSS = e_1^2 + e_2^2+ e_3^2+...+ e_n^2`
- The goal is to find rectangular regions that minimize the residual sum of squares (RSS), which measures how close the predictions are to the actual responses.
- Ideal : put all **possible partitions** of the feature space into regions would be considered to **minimize the RSS**.
- However, this is **computationally infeasible**, so a top-down, greedy approach called ***recursive binary splitting*** is used.

$$\sum_{j=1}^{j} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Baseball

- RSS
- Residual(e) : ***difference*** between an **observed value** and the **predicted value** of a variable.
- `RSS = e_1^2 + e_2^2+ e_3^2+...+ e_n^2`
- The goal is to find rectangular regions that minimize the residual sum of squares (RSS), which measures how close the predictions are to the actual responses.
- Ideal : put all **possible partitions** of the feature space into regions would be considered to **minimize the RSS**.
- However, this is **computationally infeasible**, so a top-down, greedy approach called ***recursive binary splitting*** is used.

$$\sum_{j=1}^{j} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

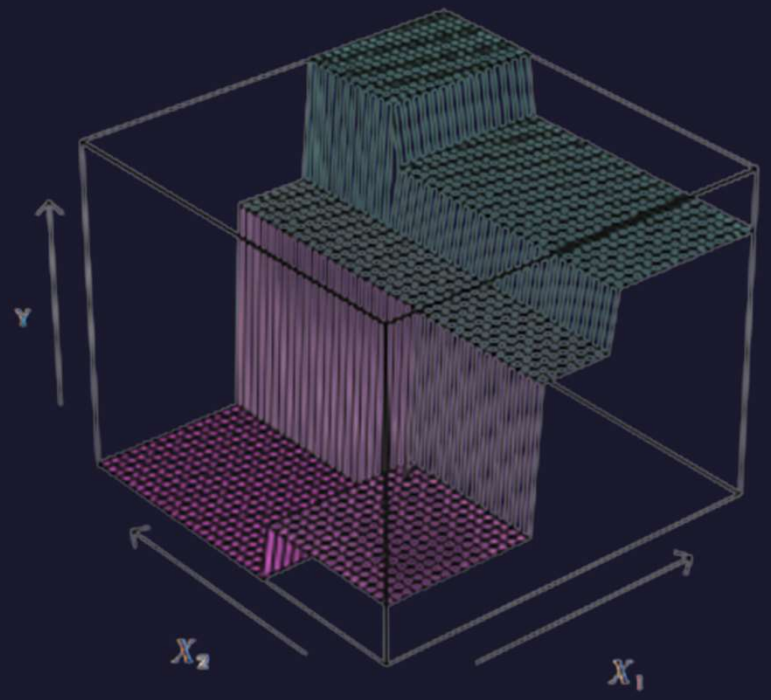
- In more detail, for each possible predictor j and split point s:
  - We define two half-planes or regions: $\{X|X_j < s\}$and$\{X|X_j >= s\}$
  - We calculate the RSS for the observations in each half-plane
  - We compare this RSS to the RSS from splits using other predictors and split points
  - The split that results in the greatest decrease in RSS (minimal RSS across the two regions) is chosen for that node in the tree.
- This process is repeated recursively at each node, selecting the predictor and split point that best splits the observations at that node into two child nodes.

$$R_1(j,s) = \{X|X_j < s\} \ \ and \ \ R_2(j,s) = \{X|X_j \geq s\}$$

and we seek the value of j and s that minimize the equation

$$\sum_{i:x_i \in R_1(j,s)} (y_i - y_{\hat{R}_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - y_{\hat{R}_2})^2$$

Tree in 3 predictive variable space

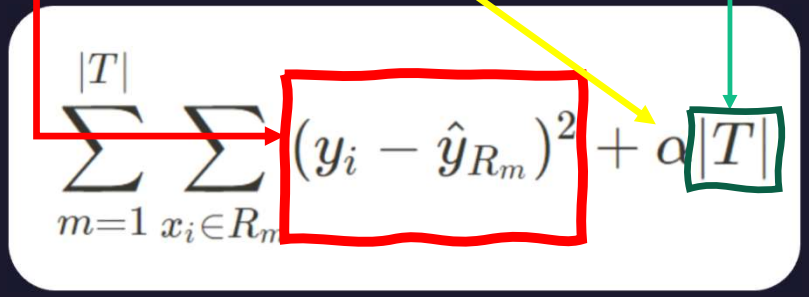



Tree pruning

- The tree generated by the recursive splitting process may perform well on the training data but poorly on new test data due to overfitting.
- This is because the tree can become too complex, leading to **high variance**.
- A simpler tree with fewer splits (regions) may have lower variance and better interpretability, at the cost of some bias.
- A good approach is to limit the size of the tree directly, for example by setting a maximum tree depth or number of terminal nodes. (Pre Pruning)
- This **balances the bias-variance tradeoff** by controlling the complexity of the tree to avoid underfitting and overfitting

- Growing a large tree and then pruning it back is a better strategy than limiting the number of splits during tree growth.(Post Pruning)
- The goal is to select a **subtree that minimizes the test error rate**.
- **cost complexity pruning** is used to select a sequence of subtrees indexed by a **tuning parameter α.** This sequence spans the range from the largest tree to the smallest tree with one terminal node.
- For a given value of α, **cost complexity pruning CCP selects** the subtree that:
  - Minimizes the empirical risk (the training error rate)
  - Has the smallest number of terminal nodes among all subtrees with the same empirical risk
- By varying α, a sequence of increasingly pruned subtrees is obtained.
- The subtree with the smallest estimated test error, based on cross-validation or a validation set, is then selected as the final tree model.
- In this way, **cost complexity pruning** provides a systematic approach to selecting a subtree that balances bias and variance for optimal test set performance

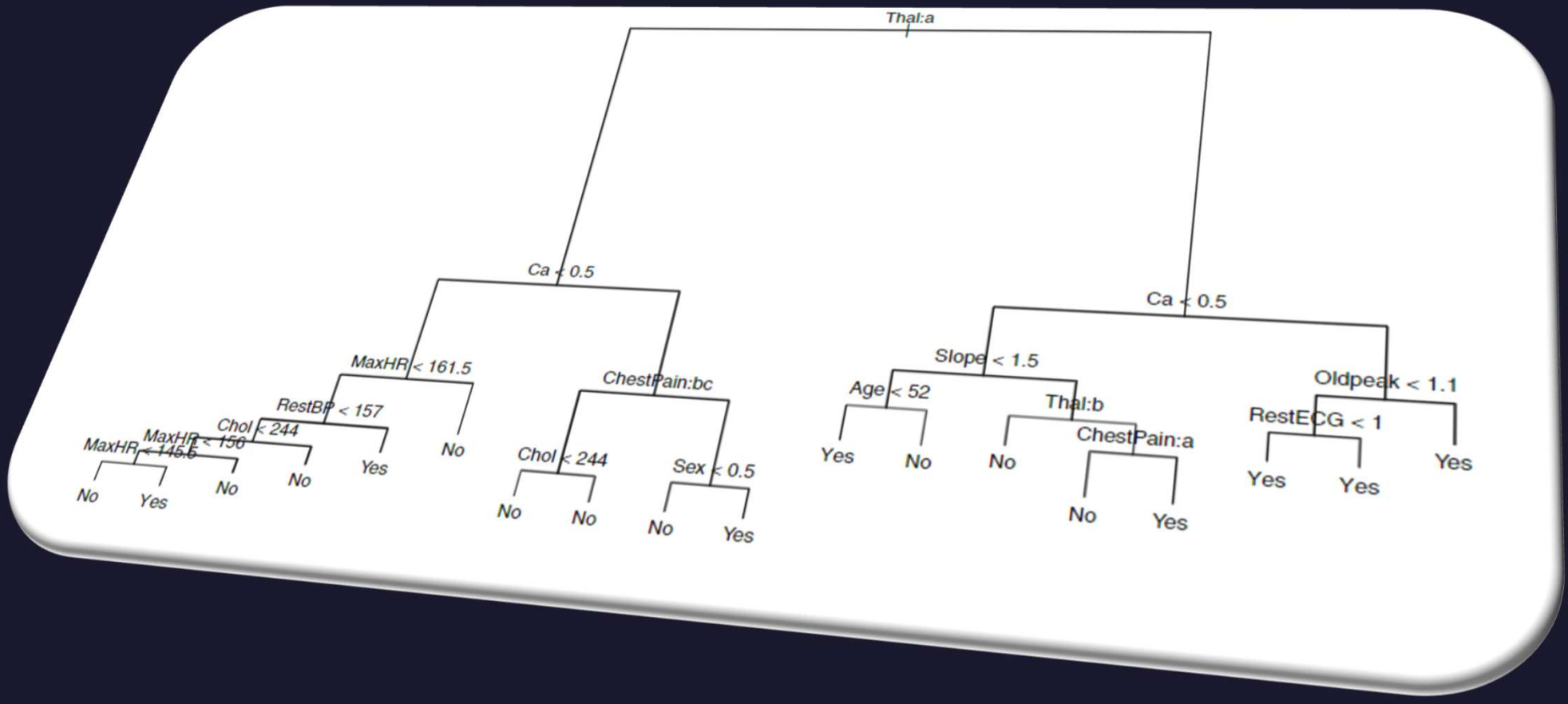
- Cost complexity pruning selects subtrees that minimize the following quantity:
- ***Training error* + *α* * *Number of terminal nodes***
- For each value of α there corresponds a subtree T ⊂T_0 such that

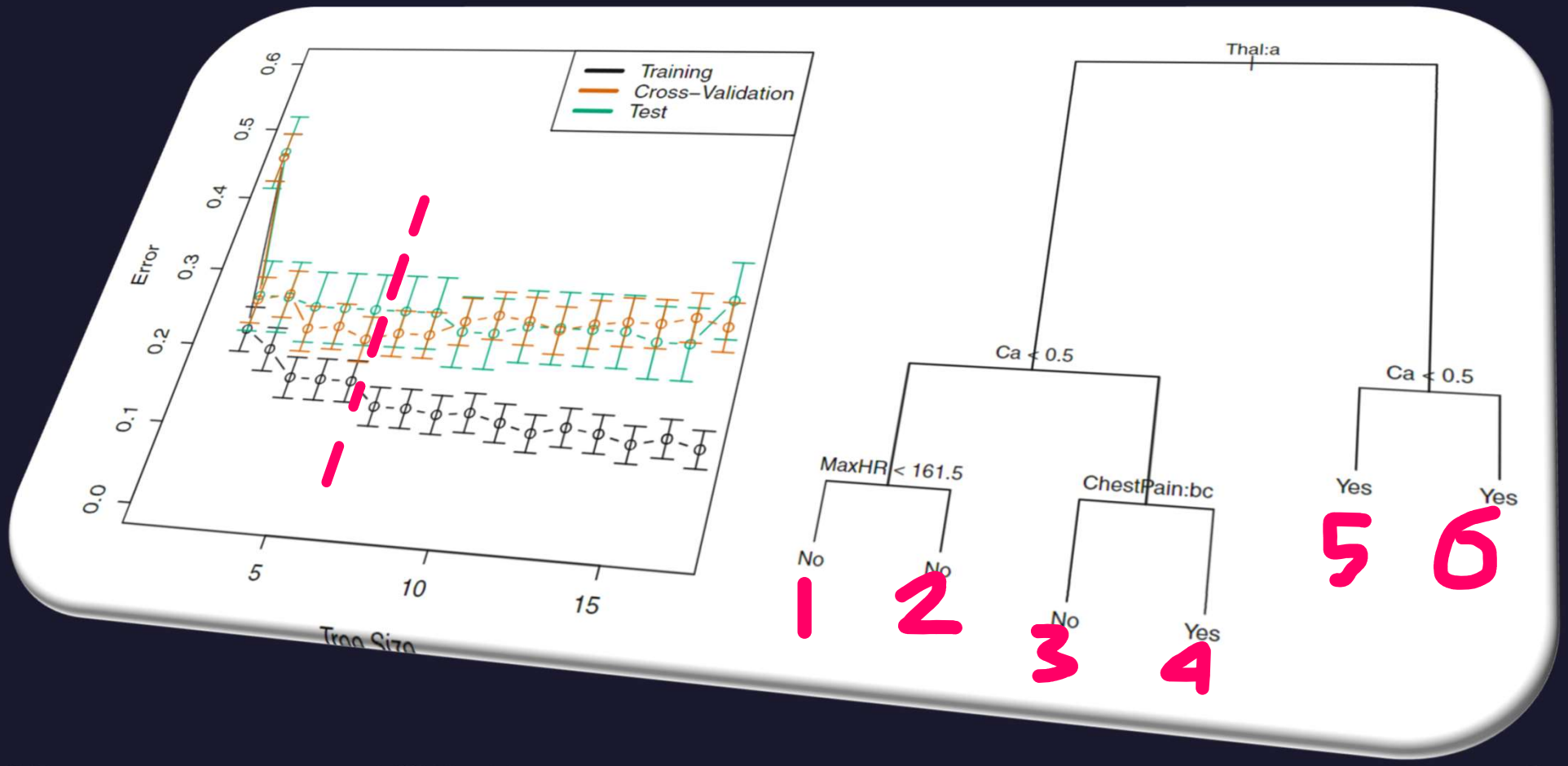


- Cost complexity pruning selects subtrees that minimize the following quantity:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- ***Training error* + *α* * *Number of terminal nodes***
- The parameter **α controls the trade-off between complexity and training error.**
- When α = 0, the subtree with the lowest training error is selected (the full tree T_0).
- As α increases, subtrees with fewer terminal nodes (smaller size) are favored to control for complexity.
- Increasing α in a nested fashion prunes branches from the tree in a predictable way.
- This produces a sequence of subtrees indexed by α, ranging from the full tree to the tree with one terminal node.


Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Slope < 1.5
Oldpeak < 1.1
RestBP < 157
Chol < 244
Age < 52
Thal:b
RestECG < 1
MaxHR < 156
MaxHR < 145.5
Chol < 244
Sex < 0.5
ChestPain:a
No
Yes
No
No
Yes
No
No
No
No
Yes
Yes
No
No
Yes
Yes
Yes
Yes
Yes
No
Yes


Training
Cross–Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
No
No
No
Yes
Yes
Yes
1
2
3
4
5
6

Thank You

See 👀

- http://www.r2d3.us/visual-intro-to-machine-learning-part-1/
- http://www.r2d3.us/visual-intro-to-machine-learning-part-2/

References

- http://www.r2d3.us/visual-intro-to-machine-learning-part-1/
- http://www.r2d3.us/visual-intro-to-machine-learning-part-2/
- https://hossam-ahmed.notion.site/8-Tree-based-Model-c3a1186914ed40f7b4298dd5493f3fb3?pvs=4
- https://hossam-ahmed.notion.site/Entropy-Information-Gain-Gini-Index-1fbaf1424fe8495f91b68d11a071a930?pvs=4
- https://en.wikipedia.org/wiki/Decision_tree_learning
- Book: statistical learning