INTRO TO RANDOM FOREST

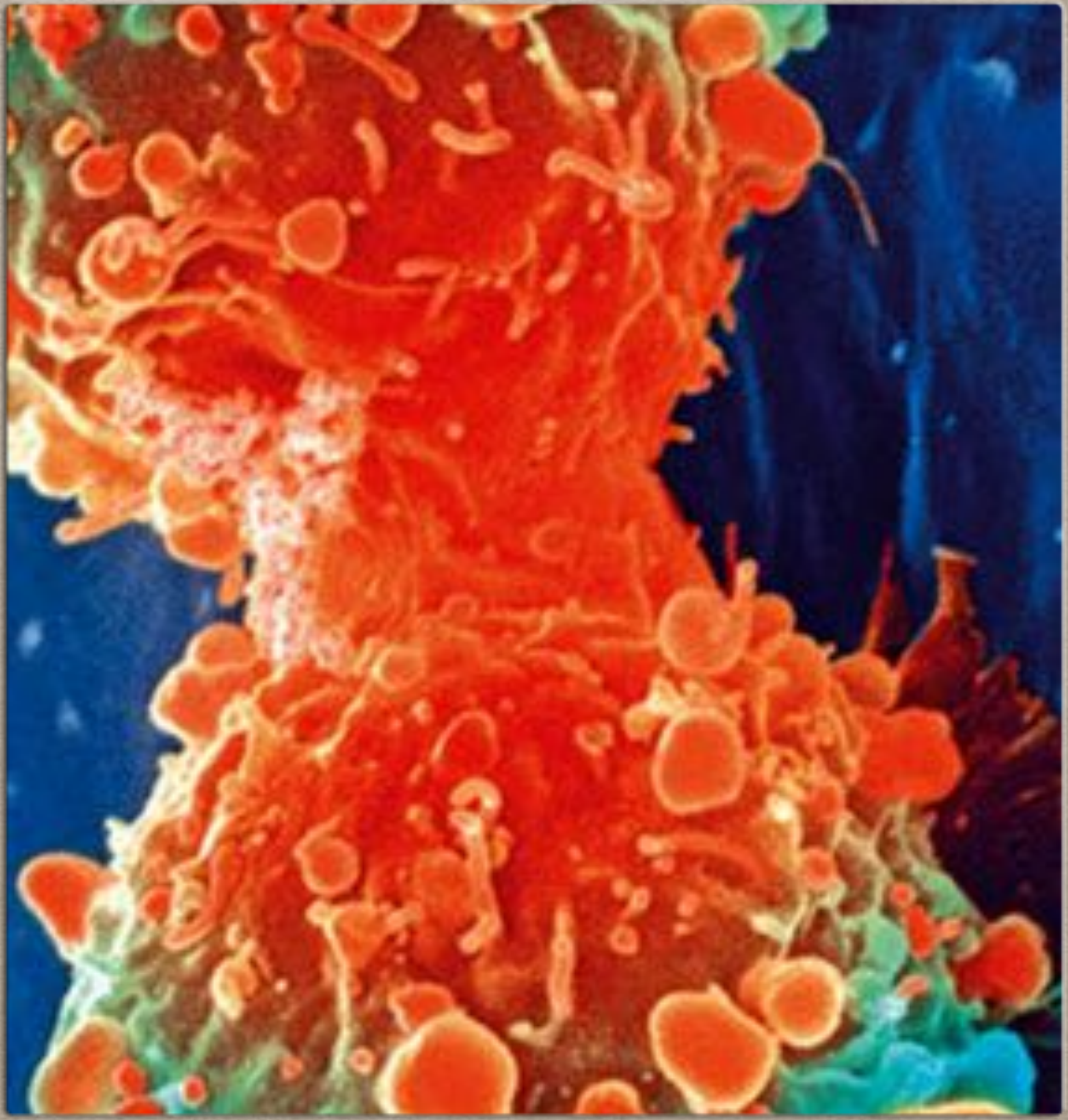
BY ANTHONY ANH QUOC DOAN

MOTIVATION FOR RANDOM FOREST

- Random forest is a great statistical learning model.
- It works well with small to medium data. Unlike Neural Network which requires large data to train. (1)
- Example: Ted's Cancer Thesis
- source: "An Empirical Comparison of Supervised Learning Algorithms" - https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

EXAMPLE: PREDICTING CANCER

- Clinical data have few observations ranging from 5 patients to 200 patients (for phase 3).
- Medical data cost money. Very expensive to run trials and to find patient willing to volunteer, especially rare disease.
- Of course 5 is small but 200 is great for random forest. Bad for Neural Network.
- Ted's Thesis

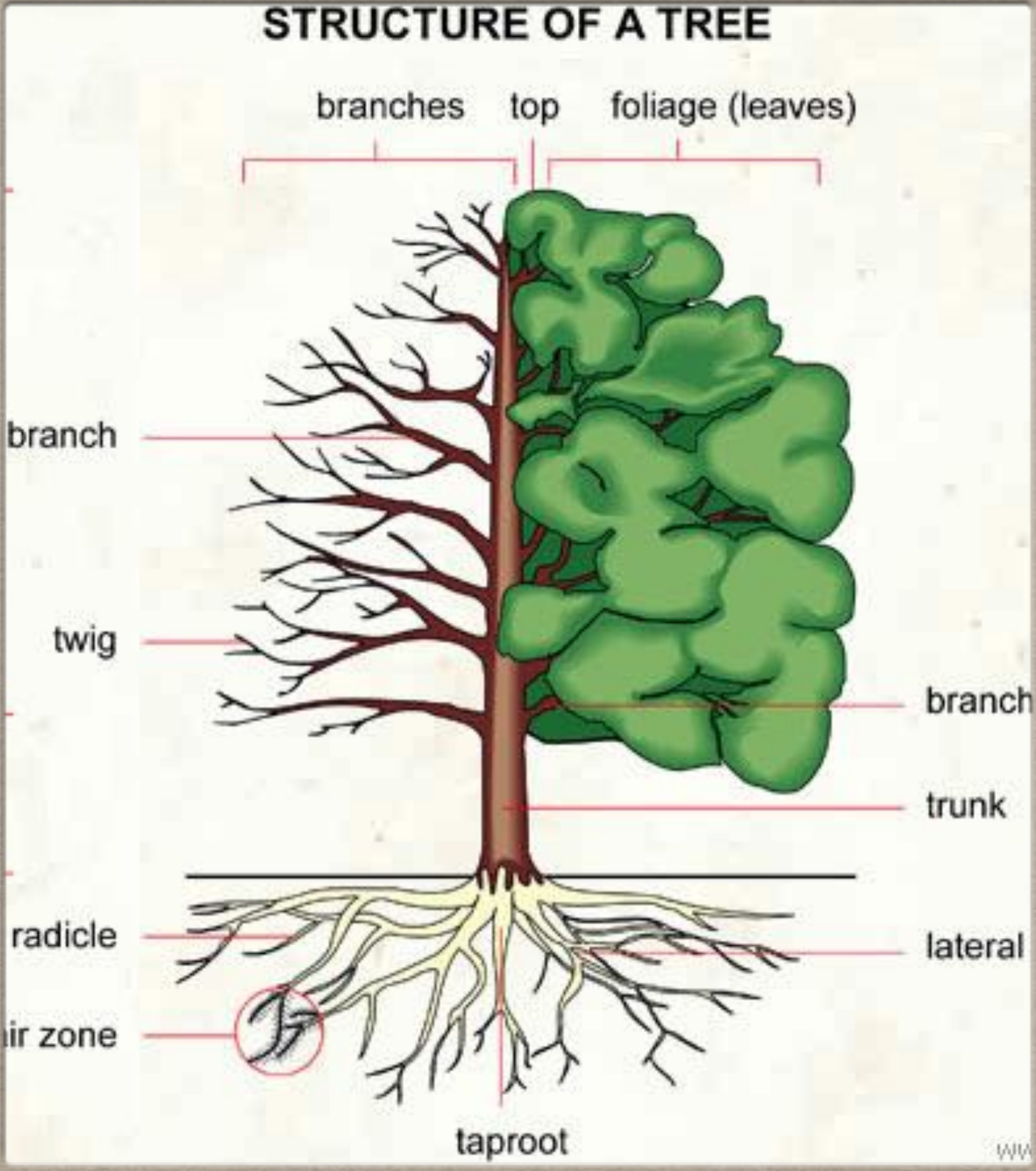


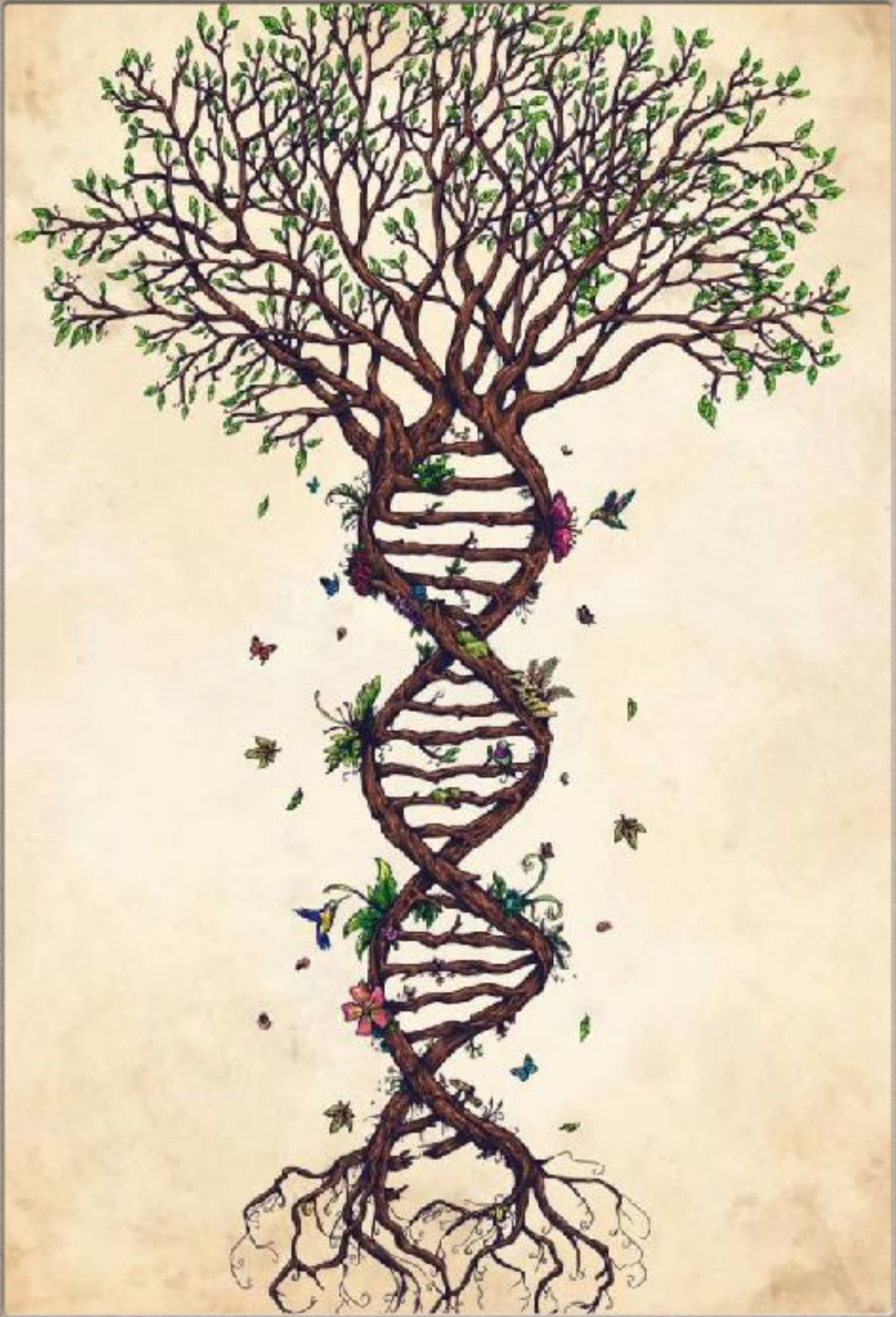
ROADMAP SECTIONS

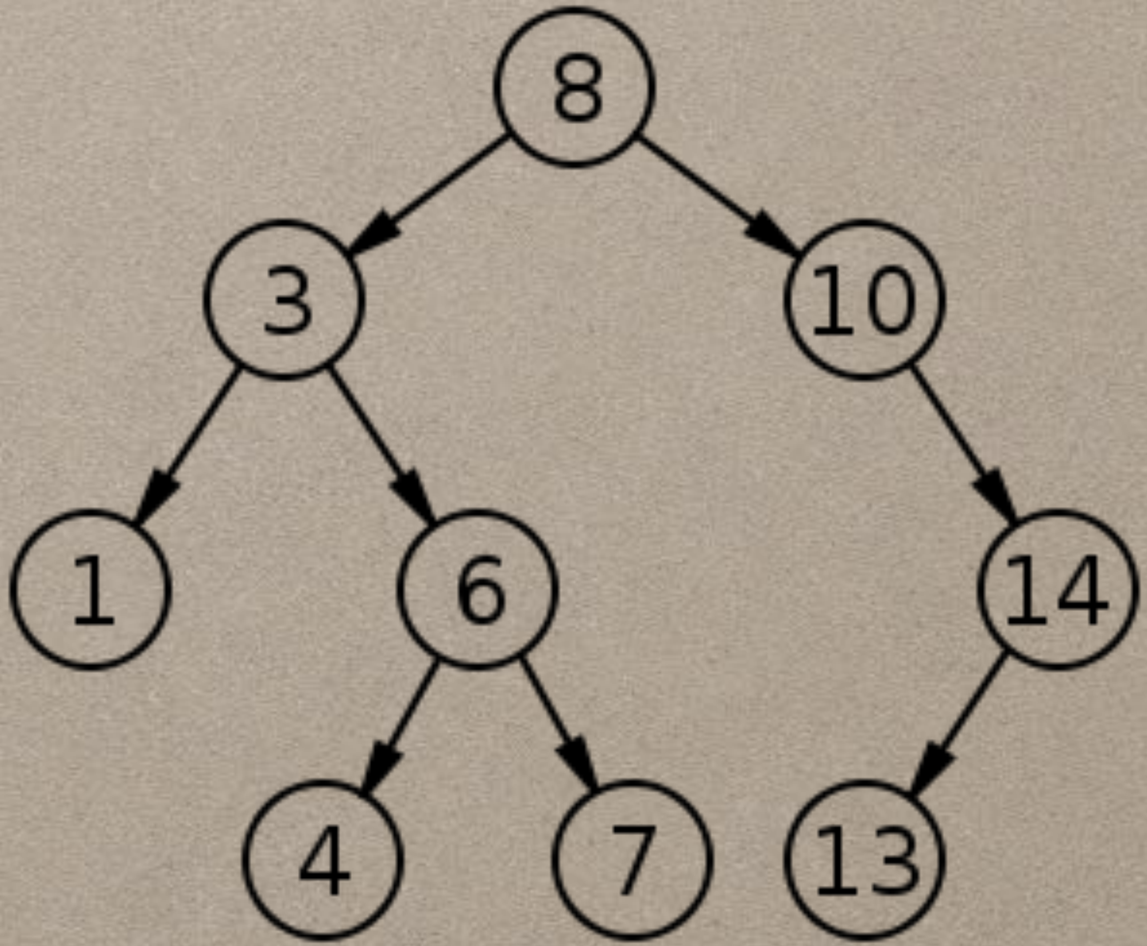
1. CART (CLASSIFICATION AND REGRESSION TREE)

CART
SECTION OVERVIEW

1. History
2. Quick Example
3. CART Overview
4. Recursive Data Partition (CART)
5. Classifying with Terminal Nodes
6. Basis Function
7. Node Splitting
8. Tree Pruning
9. Regression Tree



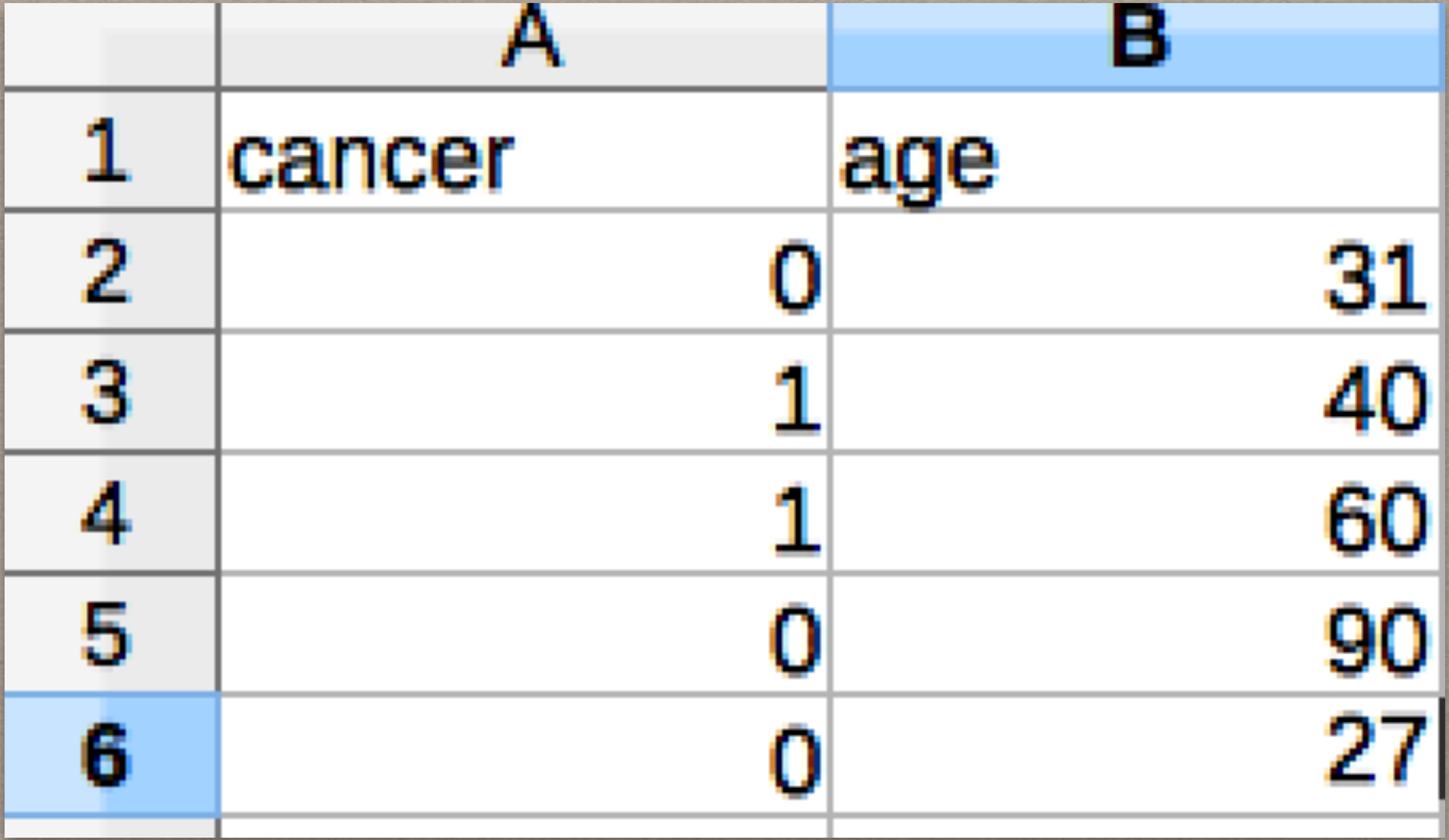



8
3
10
1
6
14
4
7
13

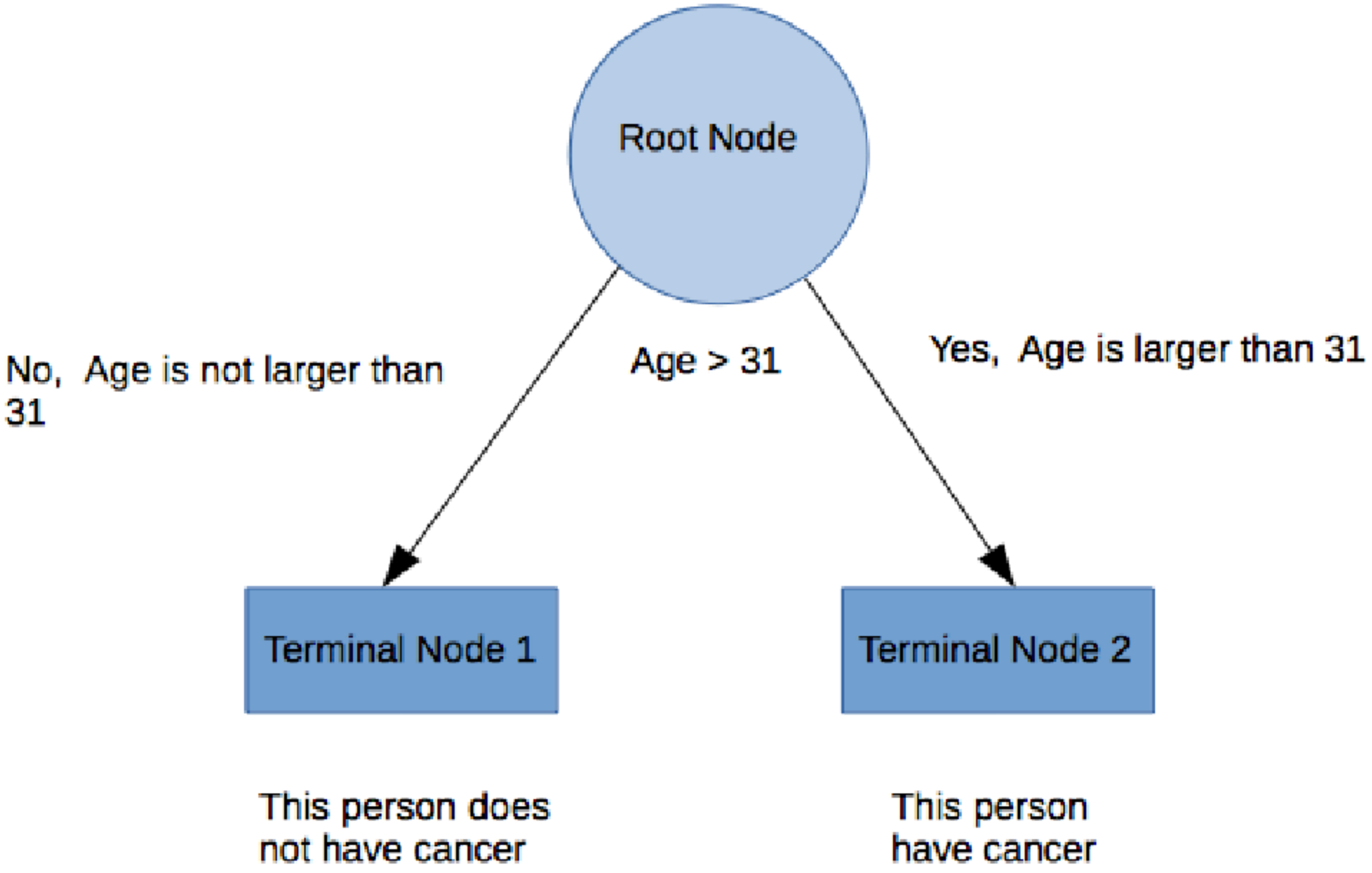
CLASSIFICATION TREE - (BINARY TREE DATA STRUCTURE)

CLASSIFICATION TREE - QUICK EXAMPLE

- We have several predictors (x's) and one response (y, the thing we're trying to predict)
- Let's do cancer with a simple model.
    1. response Y is either 0,1 where the person have cancer or not
    2. predictor - age (value range: 0 year old to 99 year old)



	A	B
1	cancer	age
2	0	31
3	1	40
4	1	60
5	0	90
6	0	27


Root Node
No, Age is not larger than 31
Age > 31
Yes, Age is larger than 31
Terminal Node 1
Terminal Node 2
This person does not have cancer
This person have cancer

HOW CART (DATA PARTITION ALGORITHM) WORKS

1. For each predictor, all possible splits of the predictor values are considered
2. For each predictor, the best split is selected. (we'll get to best split criteria)
3. With the best split of each predictor is determined, we pick the best predictor in that group.

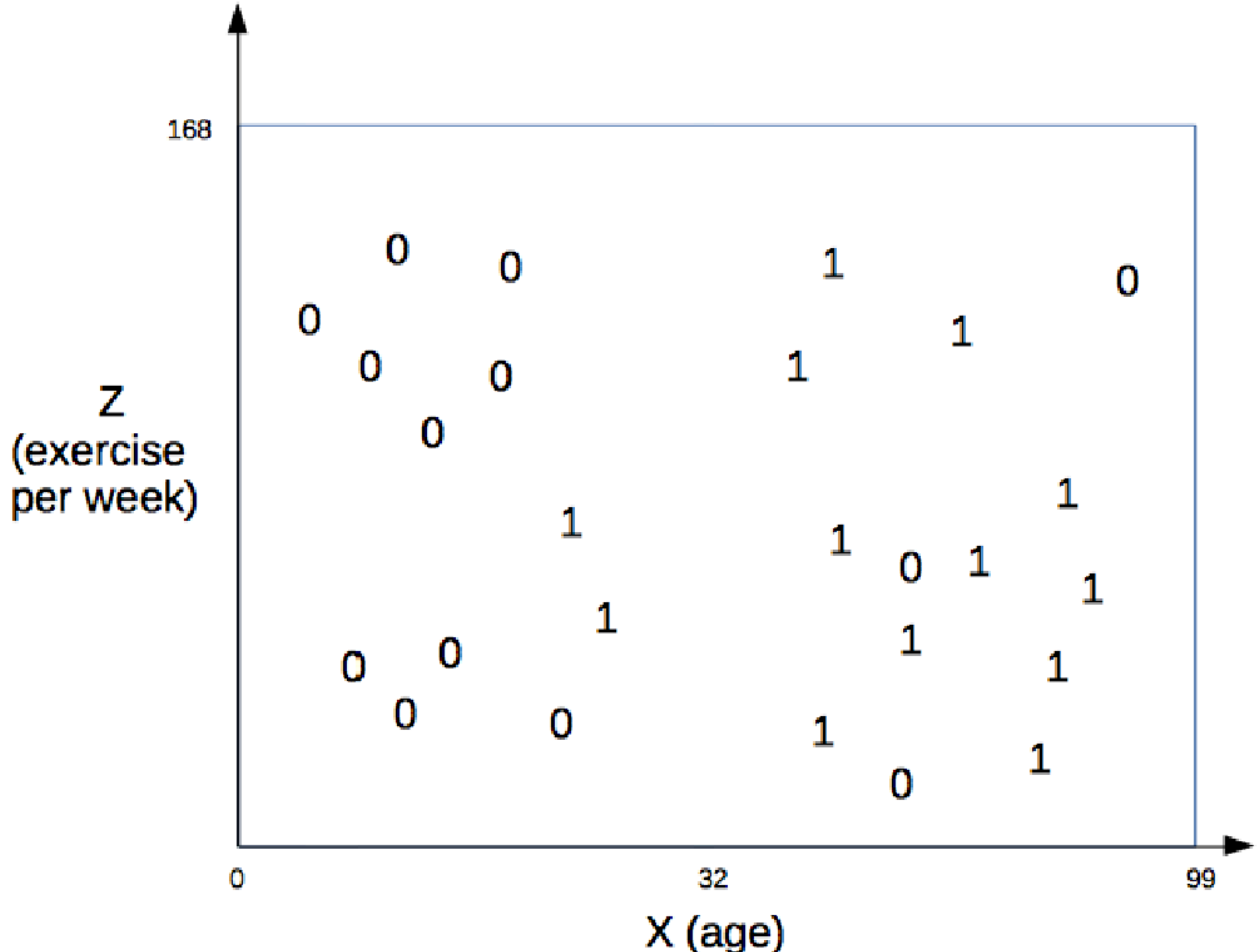
CART Uses Binary Tree Data Structure

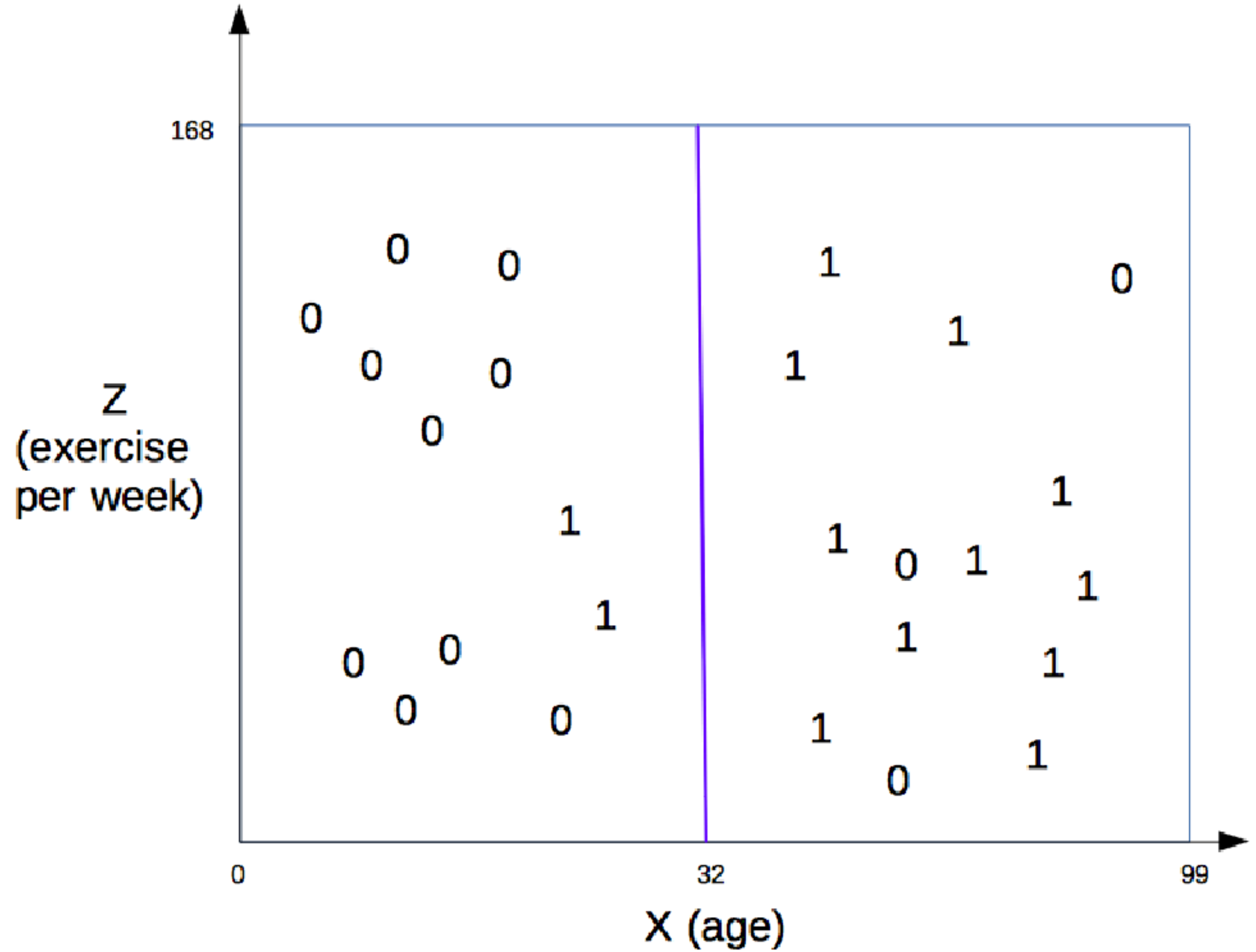
CART - DATA PARTITION EXAMPLES

- Examples of step 1:
  - marital status (categorical data: never married, married, and divorced)
    - never married vs [married and divorced]
    - married vs [never married and divorced]
    - divorced vs [never married and married]
  - age (value range: 21 to 24)
  - So the possible split (maintaining order)
    - 21 vs 22-24
    - 21-22 vs 23-24
    - 21-23 vs 24

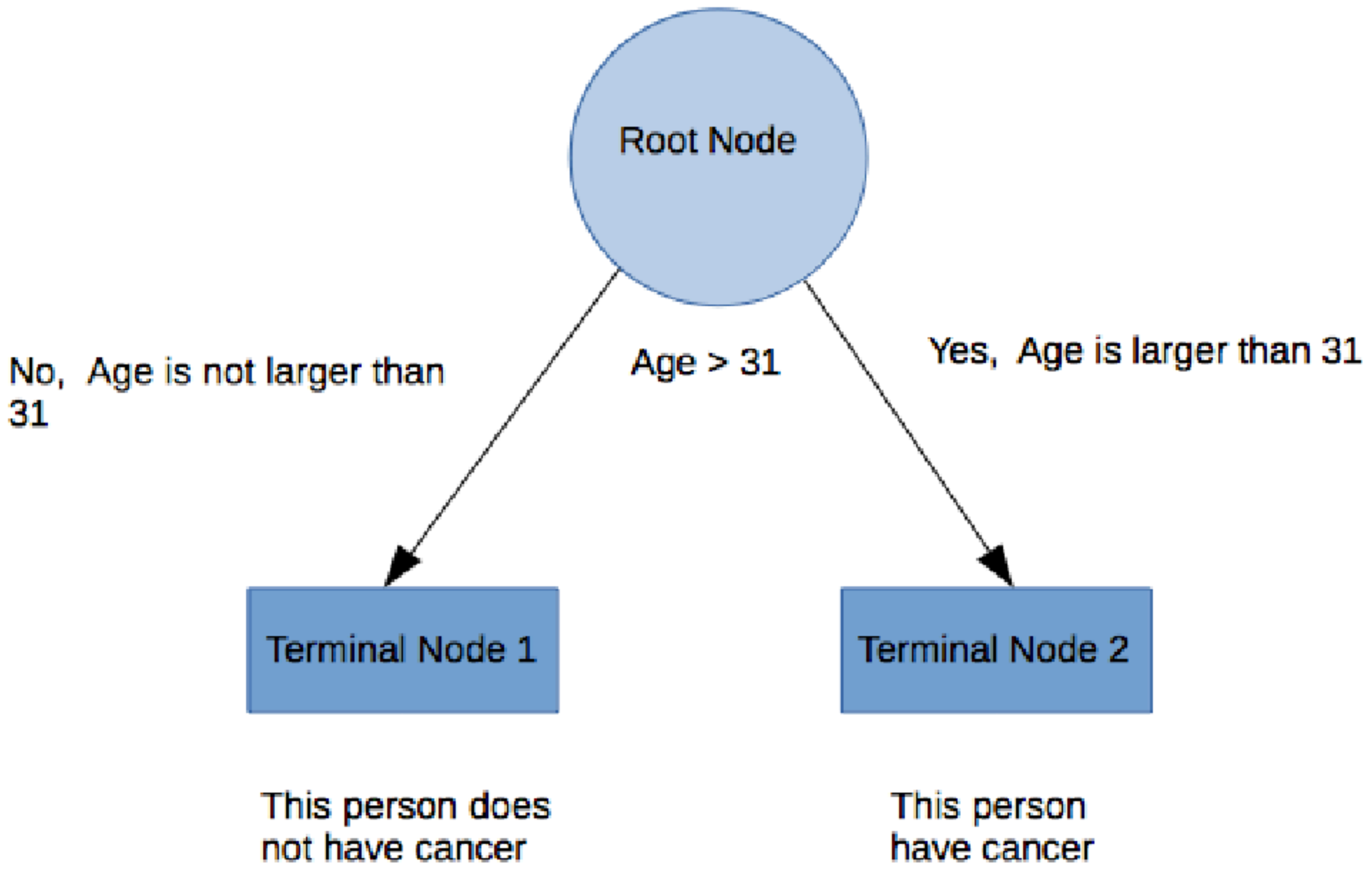
VISUAL EXAMPLE

- The best way to do this is adding another predictor to our model.
- We're going to add exercise per week (hours)
- recap our response:
  - **y** (0 - no cancer or 1- cancer)
- our predictors:
  - **age** (value range: 0 to 99 year)
  - **exercise per week** (value: 0 to 168 hours).


168
Z
(exercise
per week)
0
32
99
X (age)

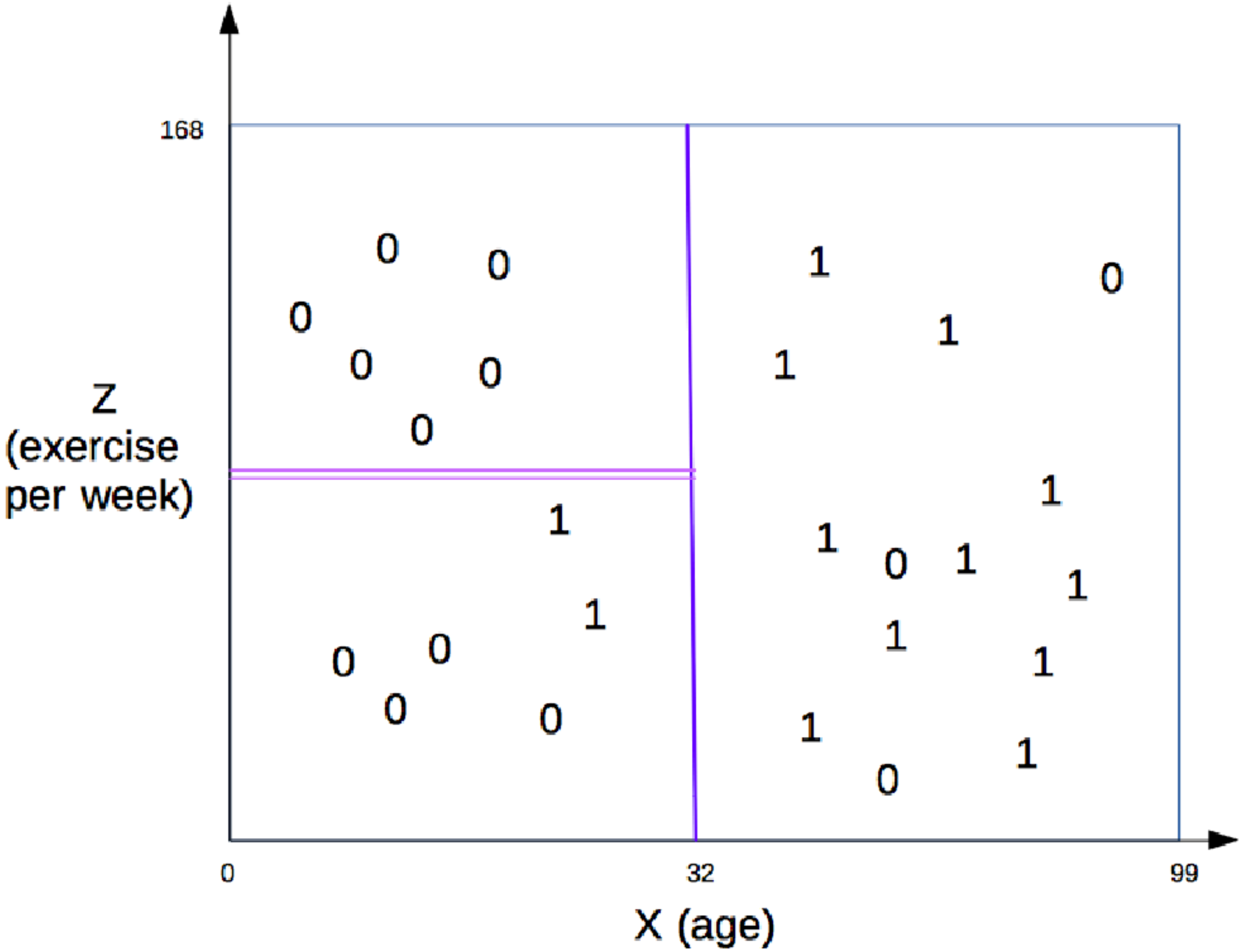

168
Z
(exercise per week)
0
32
99
X (age)

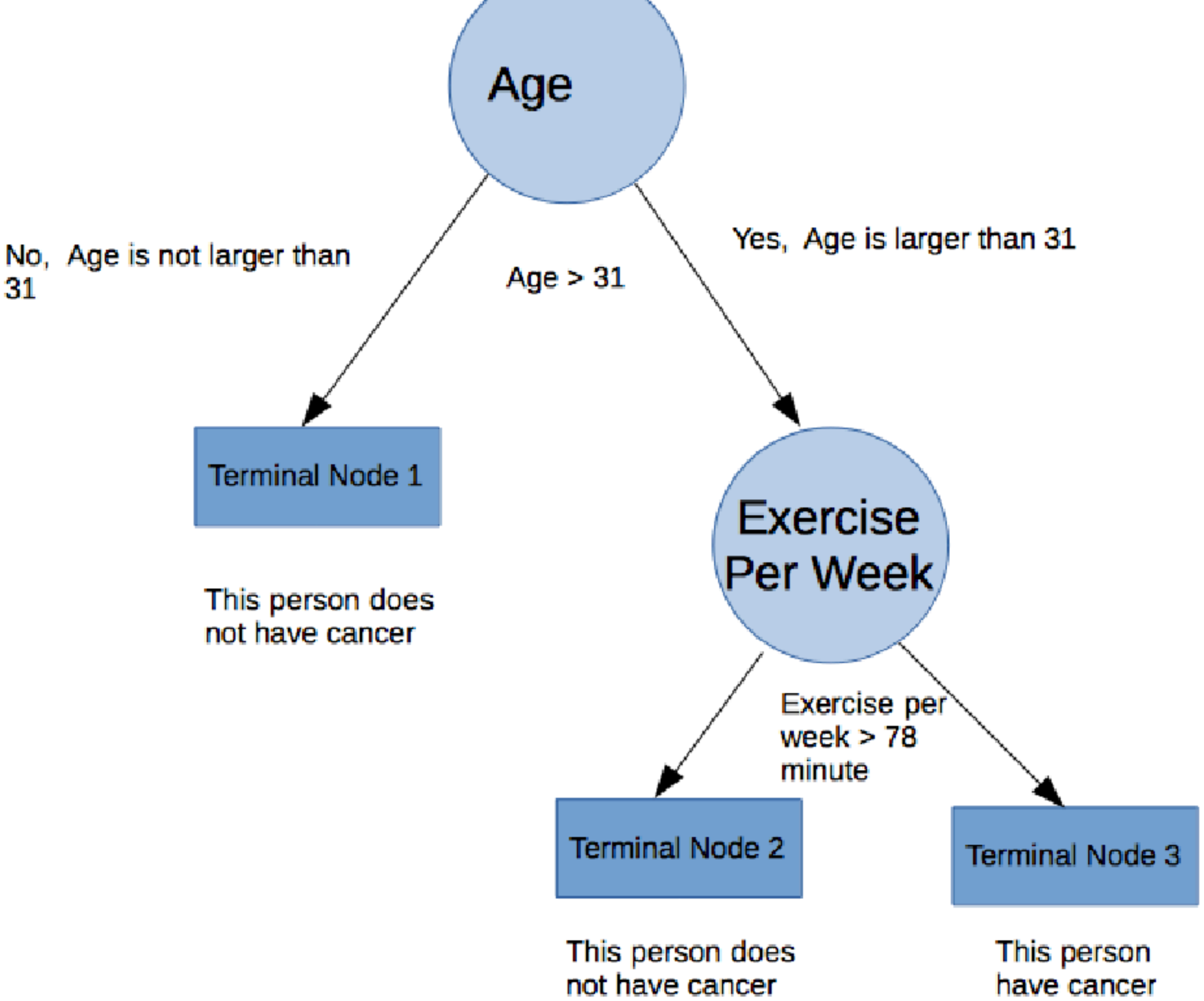
Note here we can either partition the data horizontally or vertically. We're choosing the best split/partition.


Root Node
Age > 31
No, Age is not larger than 31
Yes, Age is larger than 31
Terminal Node 1
Terminal Node 2
This person does not have cancer
This person have cancer

REPEAT DATA PARTITION ALGORITHM, CART, AGAIN FOR THE PARTITIONED DATA

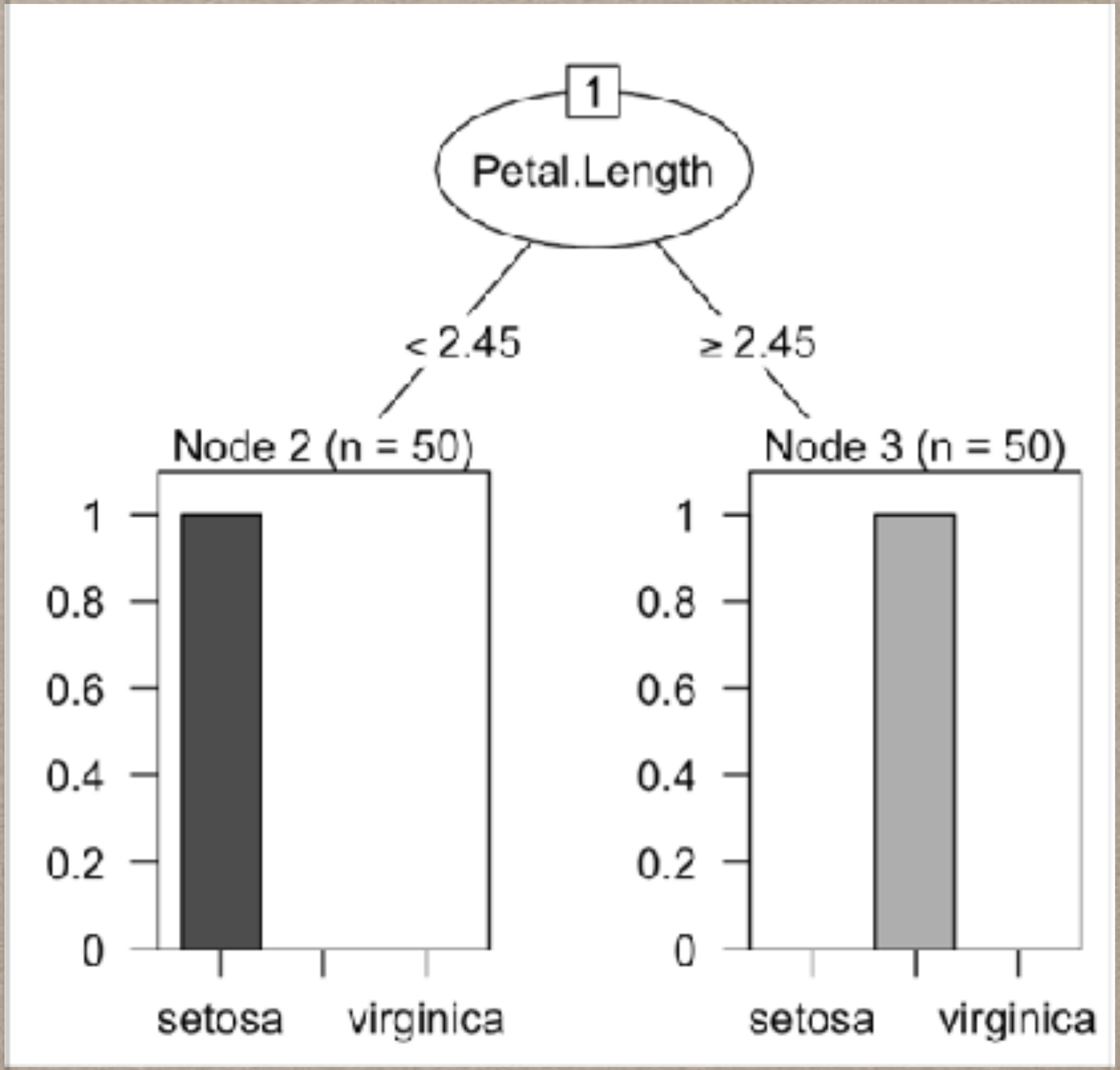
RECURSIVELY…


168
Z (exercise per week)
0
32
99
X (age)


Age
No, Age is not larger than 31
Age > 31
Yes, Age is larger than 31
Terminal Node 1
This person does not have cancer
Exercise Per Week
Exercise per week > 78 minute
Terminal Node 2
This person does not have cancer
Terminal Node 3
This person have cancer

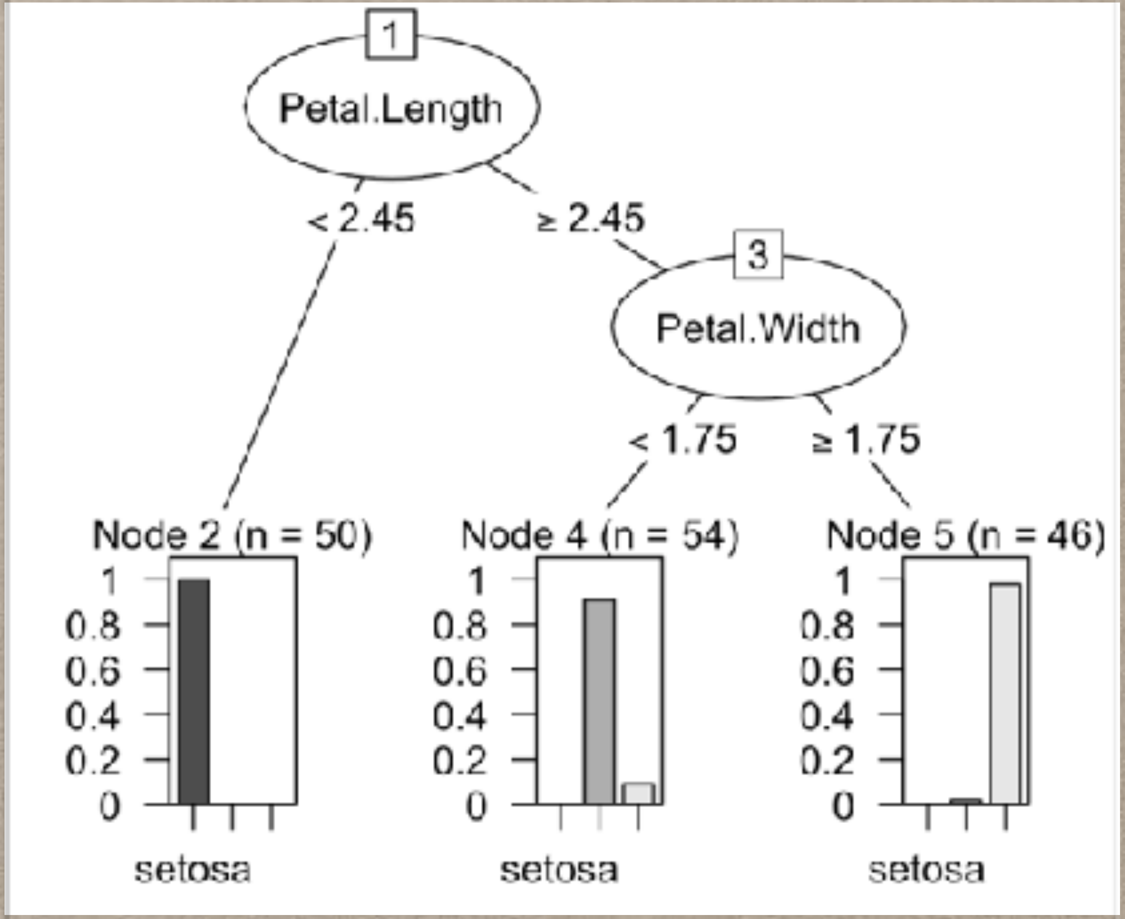
R - CODE

```
flowers_data <- iris

flowers_data <- flowers_data[!
(flowers_data$Species ==
'virginica'),]

library(rpart)

tree.model <- rpart(Species ~ . ,
data = flowers_data)

library(party)

library(partykit)

tree.model.party <-
as.party(tree.model)

plot(tree.model.party)
```



R - CODE

```
flowers_data <- iris

library(rpart)

tree.model <- rpart(Species
~ . , data = flowers_data)

library(party)

library(partykit)

tree.model.party <-
as.party(tree.model)

plot(tree.model.party)
```



$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X).$$

BASIS FUNCTIONS

- X is a predictor
- M transformations of X
- β_m is the weight given to the mth transformation (the coefficient)
- h_m is the m-th transformation of X
- f(x) is the linear combination of transformed values of X

$$f(X) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3.$$

BASIS FUNCTION EXAMPLE

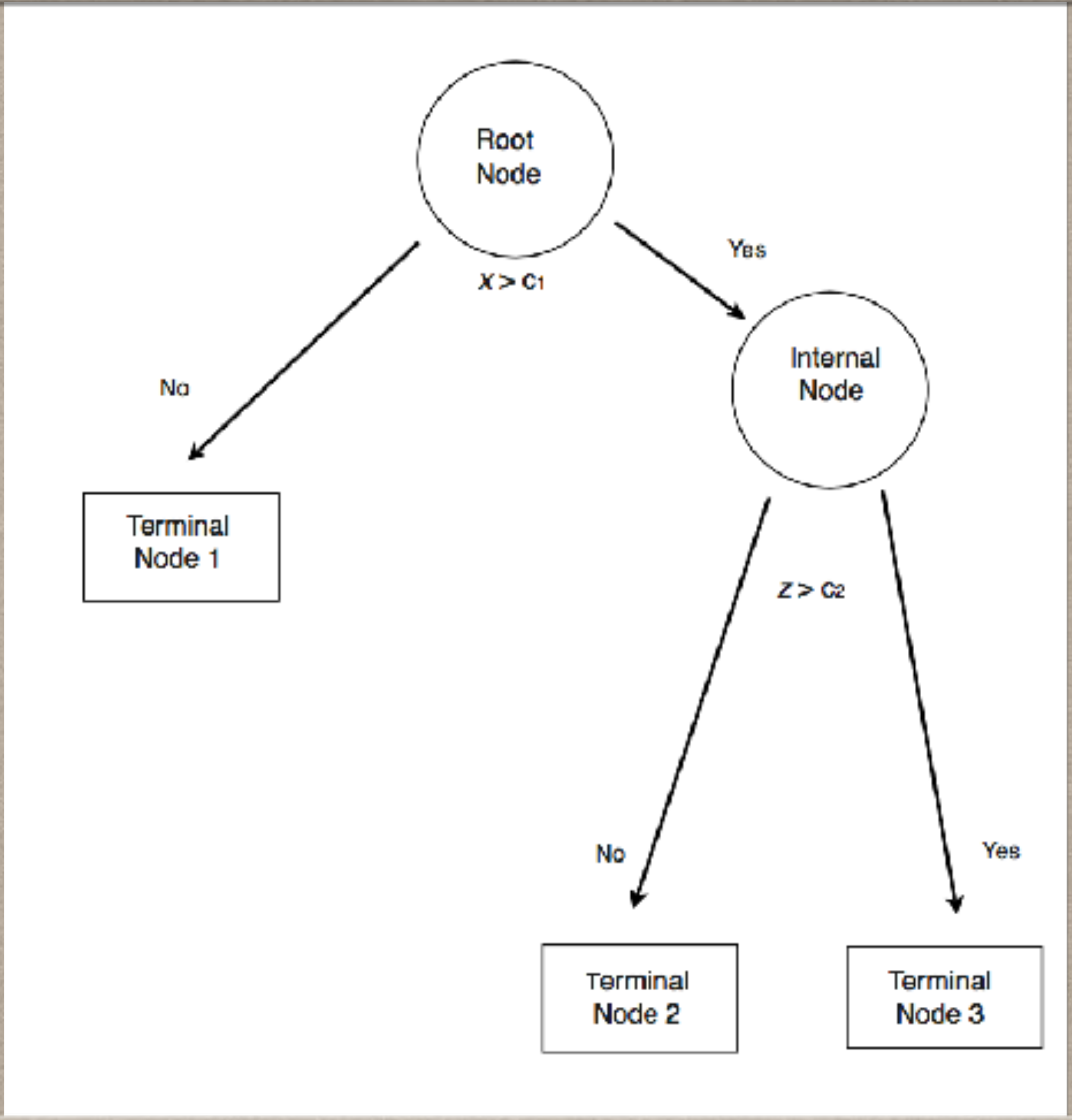
$$f(Z) = \beta_0 + \beta_1(I[z > 5]) + \beta_2(I[z > 8|z > 5]) + \beta_3(I[z < 2]).$$

THE BASIS EXAMPLE WE CARE ABOUT

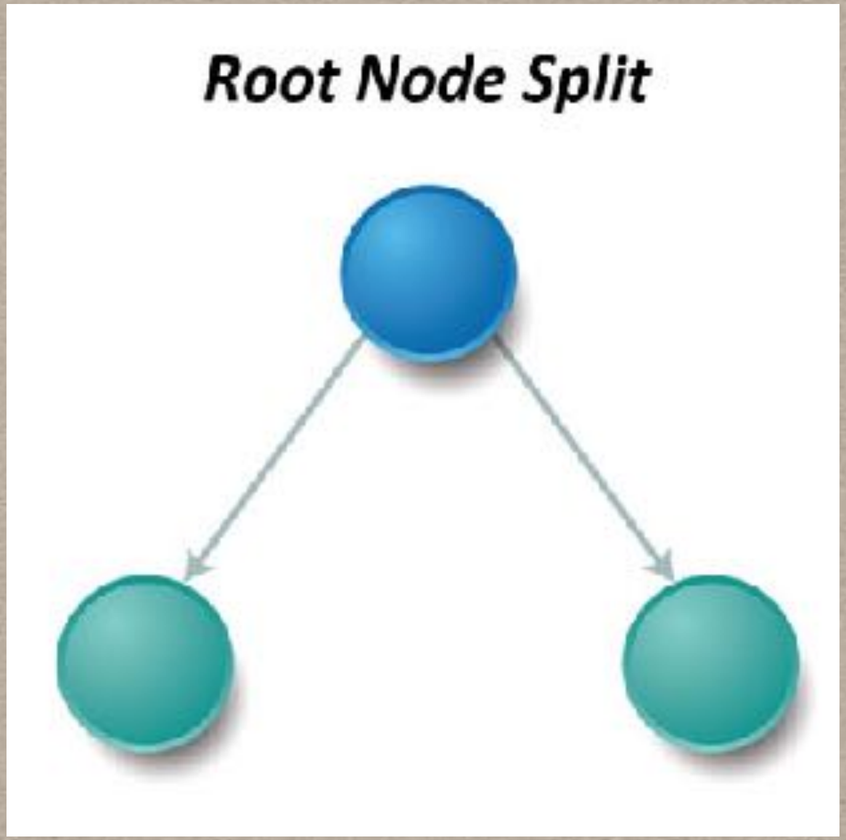
$$f(X) = \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X).$$

THIS IS CART BASIS FUNCTION

The new summation is for multiple predictors. P = total number of predictors.



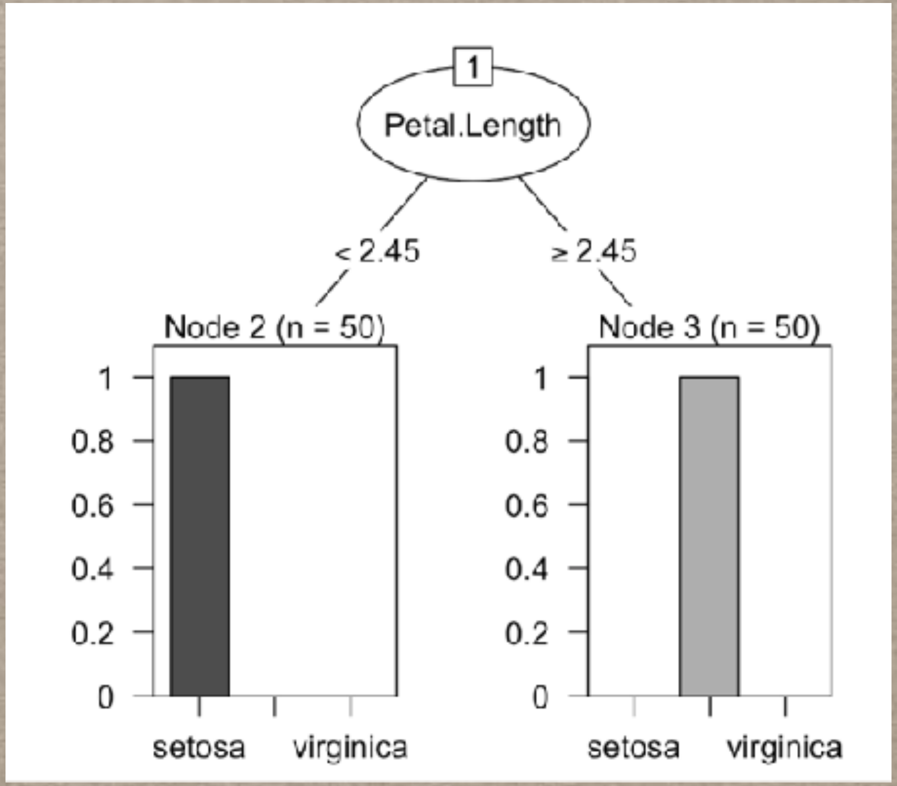
$$f(X,Z) = \beta_0 + \beta_1[(I(x \le c_1)] + \beta_2[I(x > c_1 \ \& \ z \le c_2)] + \beta_3[I(x > c_1 \ \& \ z > c_2)]$$



NODE SPLITTING

QUESTION HOW IS SPLIT DETERMINED?

- The goal is to split/partition the data until each node is homogenous in data, or as little "impurity" (few outcomes that we want in the particular node and mostly outcome that we want).



HOW IS NODE IMPURITY CALCULATED?

NODE IMPURITY - SETUP

- Our data to train is a random sample from a well defined population
- given a node, node A
- $p(y = 1 | A)$
    - impurity of node A is the probability that $y = 1$ given it is node A.

IMPURITY FUNCTION

$$I(A) = \phi[p(y = 1|A)]$$

with $\phi \geq 0$, $\phi(p) = \phi(1-p)$, and $\phi(0) = \phi(1) < \phi(p)$

- I(A) represent Impurity function that takes in a node as our parameter.
- The restriction tells us that the impurity function is nonnegative, symmetric when A contains all 0s or 1s, and a maximum of half of each (coin toss).

IMPURITY - DEFINING Φ (PHI)

$$I(A) = \phi[p(y = 1|A)]$$

- There are several Φ functions that people uses.
- The most commonly use is the Gini Index: $\Phi(p) = p\,(1-p)$
- Others
    - Bayes Error: $\Phi(p) = \min(p, 1-p)$
    - Cross Entropy Function: $\Phi(p) = -p \log(p) - (1-p) \log(1-p)$

GINI INDEX

$$I_G(f) = \sum_{i=1}^{J} f_i(1-f_i) = \sum_{i=1}^{J}(f_i - {f_i}^2) = \sum_{i=1}^{J} f_i - \sum_{i=1}^{J} {f_i}^2 = 1 - \sum_{i=1}^{J} {f_i}^2$$

$$\mathrm{Gini}(t) \quad = \quad 1 - \sum_{i=0}^{c-1}[p(i|t)]^2,$$

- **p(i|t)** denote the fraction of records belonging to class ***i*** at a given node ***t***
- ***c*** is the number of classes (example: cancer, no cancer)

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  – Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
  – Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$\boxed{GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)}$$

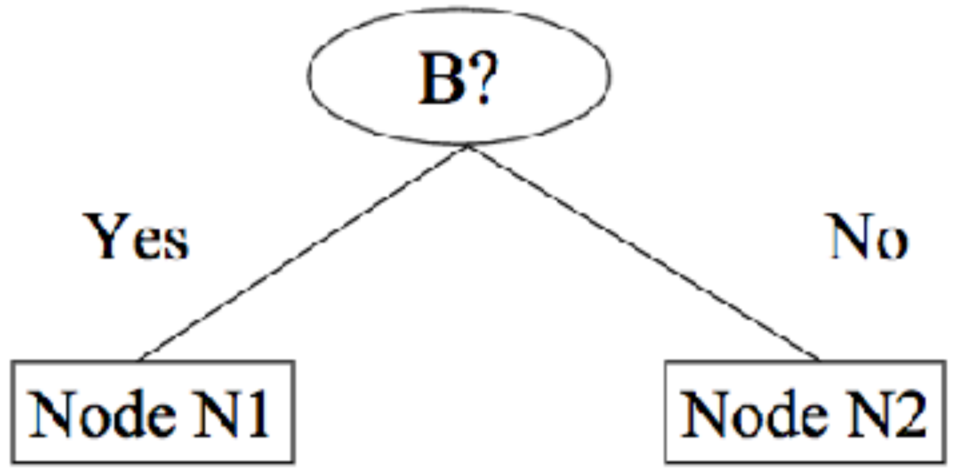
where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.

	Parent
C1	6
C2	6
Gini = 0.500	



Gini(N1)
$= 1 – (5/6)^2 – (2/6)^2$
= 0.194

Gini(N2)
$= 1 – (1/6)^2 – (4/6)^2$
= 0.528

	N1	N2
C1	5	1
C2	2	4
Gini=0.333		

Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

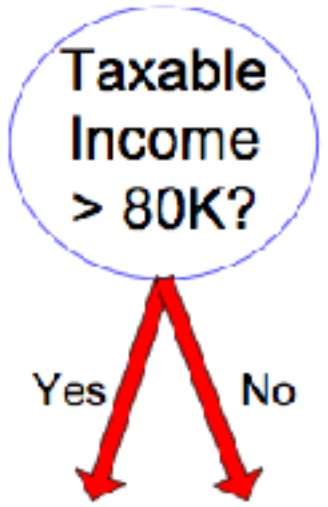
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



PRUNING TREE

WHY PRUNE?

- Reduce complexity of the model
- Reduce overfitting.
- Note: Random Forest doesn't Prune trees so we're not going to go into Tree pruning.

REGRESSION TREE

REGRESSION TREE - SPLITTING NODE

$$i(\tau) = \sum (y_i - \bar{y}(\tau))^2$$

- The only difference is the impurity function of the node
- Which is just a within-node sum of squares for the response
- Where the summation of all cases in node τ minus the mean those cases squared.
- This is SSTO (sum square total) in Linear Regression

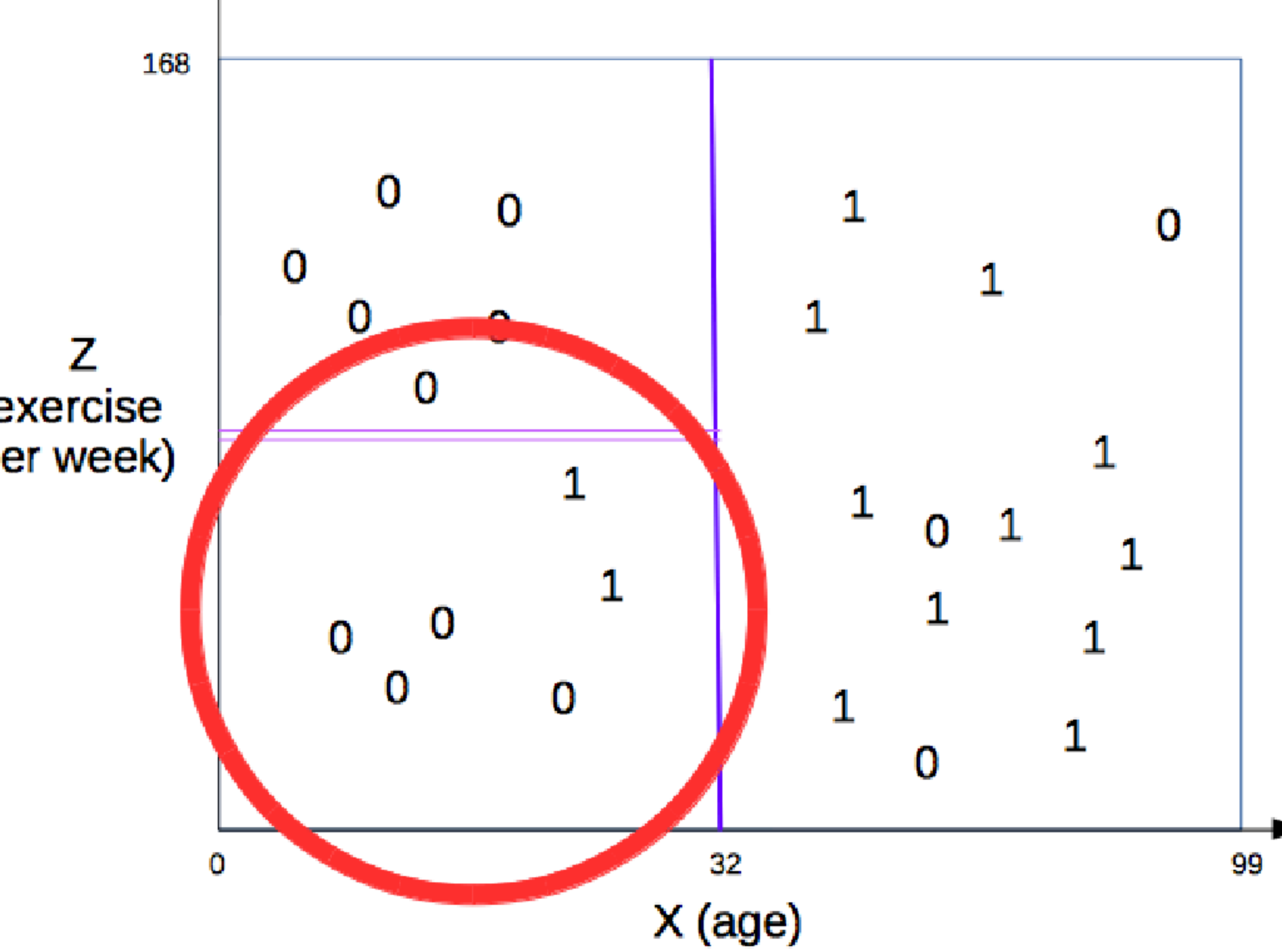
REGRESSION TREE EXAMPLE

$$i(\tau) = \sum (y_i - \bar{y}(\tau))^2$$

Warning:

We're going to do it on classification data as an example to show how to use the equation.

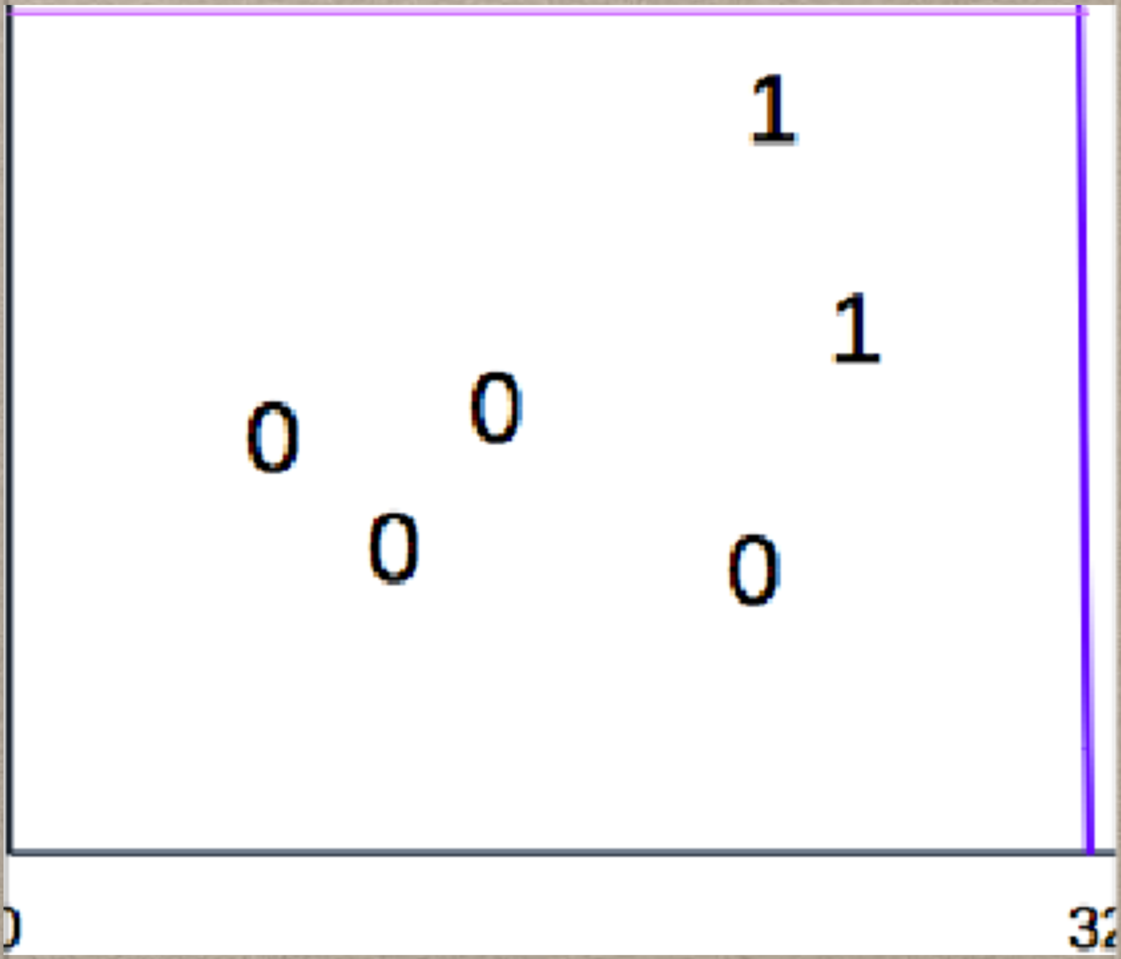
Because I don't believe I have time to go over linear regression in here to do a proper example.


168
Z
exercise
er week)
0
32
99
X (age)

$$i(\tau) = \sum (y_i - \bar{y}(\tau))^2$$

$\bar{y}(\tau)$

- = (1/n)*sum(y_i)
- = (1/6)*(0+0+0+0+1+1)
- = 2/6
- = 1/3
- y_i represents one data point in the partition (0,0,0,1,1)



REGRESSION TREE - PRUNING

- Uses AIC or some other regression criteria, it's a penalty function for using too many predictors and sacrificing degree of freedom.
- AIC and such are outside the scope of this talk because it's a huge topic and rabbit hole into linear regression.

2. BAGGING (BOOTSTRAP AGGREGATION)

BAGGING SECTION OVERVIEW

1. Benefit
2. Bootstrap Algorithm
3. Bootstrap example
4. Bagging Algorithm
5. Flaws
6. Why does this work?

WHY? BENEFITS.

- Reduce overfitting
- Reduce bias
- Break the bias-variance trade-off

BOOTSTRAP

- Before we dive into bagging algorithm we should go over bootstrap
- Bootstrap is a statistical resample method.
  - When you can't afford to get more sample (think medical data, poor grad student, cost, etc..)

BOOTSTRAP ALGORITHM

1. Used in statistic when you want to estimate a statistic of a random sample (a statistic: mean, variance, mode, etc…)
2. Using bootstrap we diverge from traditional parametric statistic, we do not assume a distribution of the random sample
3. What we do is sample our only data set (aka random sample) with replacement. We take up to the number of observations in our original data.
4. We repeat step 3 for a large number of time, B times. Once done we have B number of bootstrap random samples.
5. We then take the statistic of each bootstrap random sample and average it

BOOTSTRAP EXAMPLE

- Original data (random sample):
  - {1,2,3,1,2,1,1,1} (n = 8)
- Bootstrap random sample data:
  - {1,2,1,2,1,1,3} (n = 8); mean = 1.375
  - {1,1,2,2,3,1,1} (n = 8); mean = 1.375
  - {1,2,1,2,1,1,2} (n = 8); mean = 1.25
- The estimated mean for our original data is the mean of the statistic for each bootstrap sample (1.375+1.375+1.25)/3 = ~1.3333

BAGGING (BOOTSTRAP AGGREGATION) ALGORITHM

1. Take a random sample of size N with replacement from the data

2. Construct a classification tree as usual but do not prune

3. Assign a class to each terminal node, and store the class attached to each case coupled with the predictor values for each observation

BAGGING ALGORITHM

4. Repeat Steps 1-3 a large number of times.

5. For each observation in the dataset, count the number of times over tress that it is classified in one category and the number of times over trees it is classified in the other category

6. Assign each observation to a final category by a majority vote over the set of tress. Thus, if 51% of the time over a large number of trees a given observation is classified as a "1", that becomes its classification.

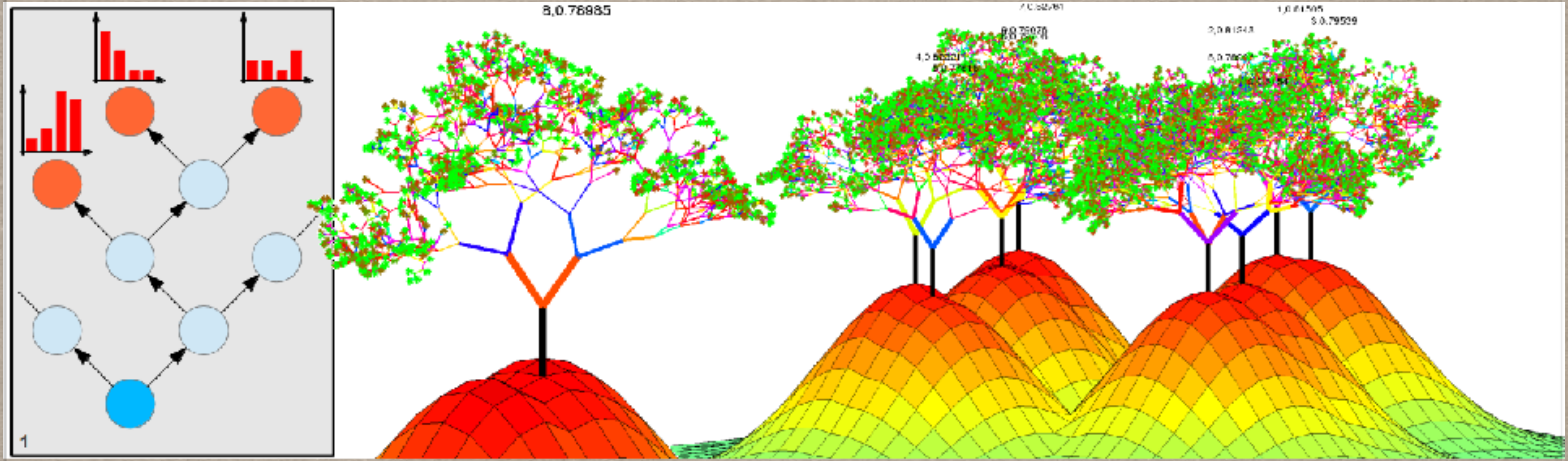
7. Construct the confusion table from these class assignments.

FLAWS

- The problem with Bagging algorithm is it's using CART.
- CART uses Gini-Index, a greedy algorithm to find the best split.
- So we end up with trees that are structurally similar to each other. The trees are highly correlated among the predictions.
- Random Forest address this.

WHY DOES THIS WORK?

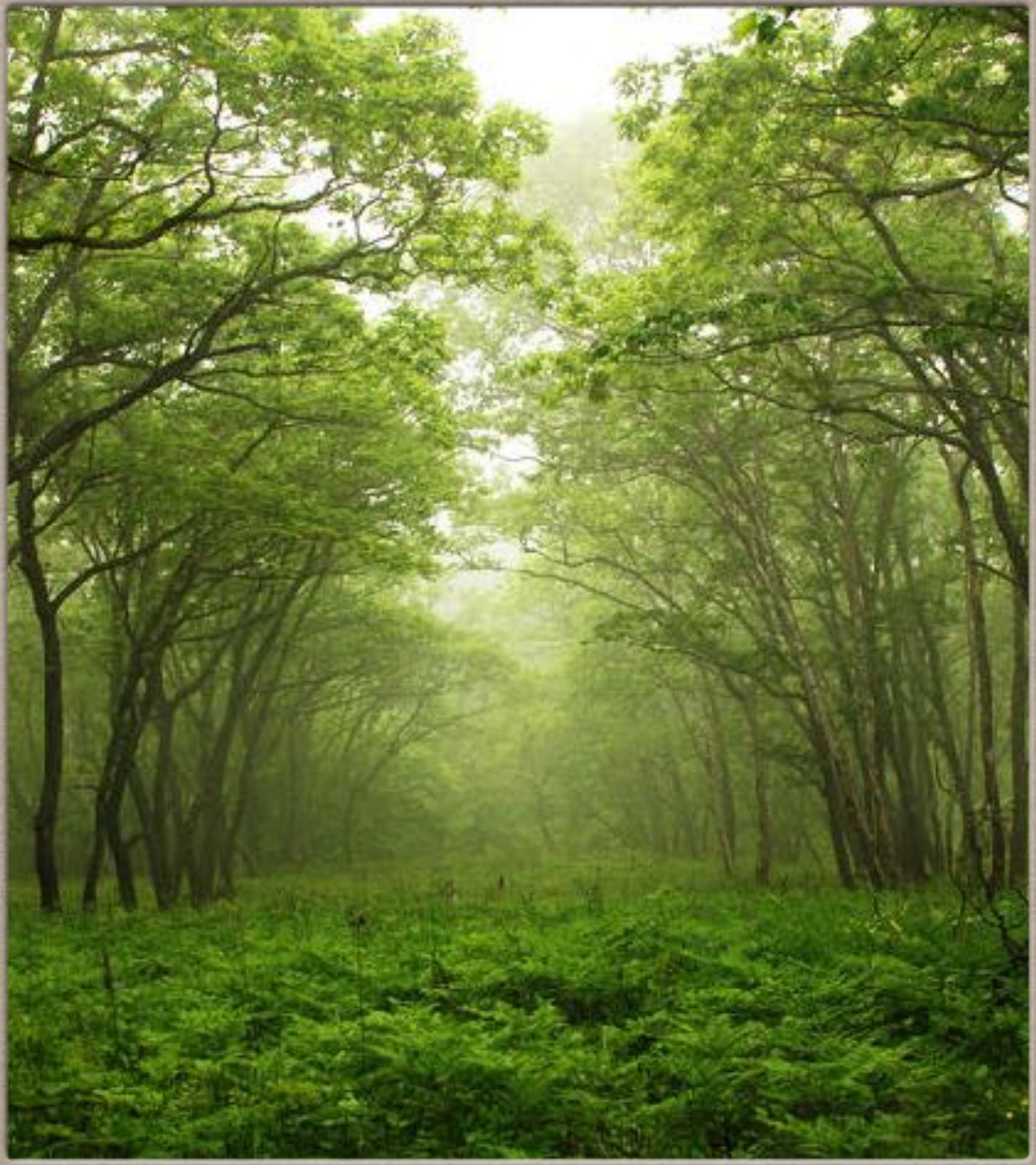
- Outside the scope of this talk.



3. RANDOM FOREST

RANDOM FOREST SECTION OVERVIEW

1. Problem RF solve
2. Building Random Forest Algorithm
3. Breaking Down Random Forest Algorithm Parts by Parts
4. How to use Random Forest to Predict
5. R Code



1. PROBLEM RANDOM FOREST IS TRYING TO SOLVE

BAGGING PROBLEM

With bagging we have an ensemble of structurally similar trees. This causes highly correlated trees.

RANDOM FOREST SOLUTION

Create trees that have no correlation or weak correlation.

2. BUILDING RANDOM FOREST ALGORITHM

RANDOM FOREST ALGORITHM

1. Take a random sample of size N with replacement from the data.

2. Take a random sample without replacement of the predictors.

3. Construct the first CART partition of the data.

BUILDING RANDOM FOREST ALGORITHM

4. Repeat Step 2 for each subsequent split until the tree is as large as desired. Do not prune.
5. Repeat Steps 1–4 a large number of times (e.g., 500).

3. BREAKING DOWN RANDOM FOREST ALGORITHM PARTS BY PARTS

1. TAKE A RANDOM SAMPLE OF SIZE N WITH REPLACEMENT FROM DATA

- This is just bootstrapping on our data (recall bagging section)

2. TAKE A RANDOM SAMPLE WITHOUT REPLACEMENT OF THE PREDICTORS

- Predictor sampling / bootstrapping
- Notice this is bootstrapping our predictors and it's <u>without replacement</u>.
- This is Random Forest solution to highly correlated trees that arises from bagging algorithm.

Question (for step 2):

Max number when sampling predictors/
features?

Answer:

2 to 3 sample for predictors/features

3. CONSTRUCT THE FIRST CART PARTITION OF DATA

- We partitioned our first bootstrap and use Gini-index on our bootstrapped predictors sample in step 2 to decided the split.

4. REPEAT STEP 2 FOR EACH SUBSEQUENT SPLIT UNTIL THE TREE IS AS LARGE AS DESIRED. DO NOT PRUNE.

- Self explanatory.

5. REPEAT STEPS 1-4 A LARGE NUMBER OF TIMES (E.G., 500).

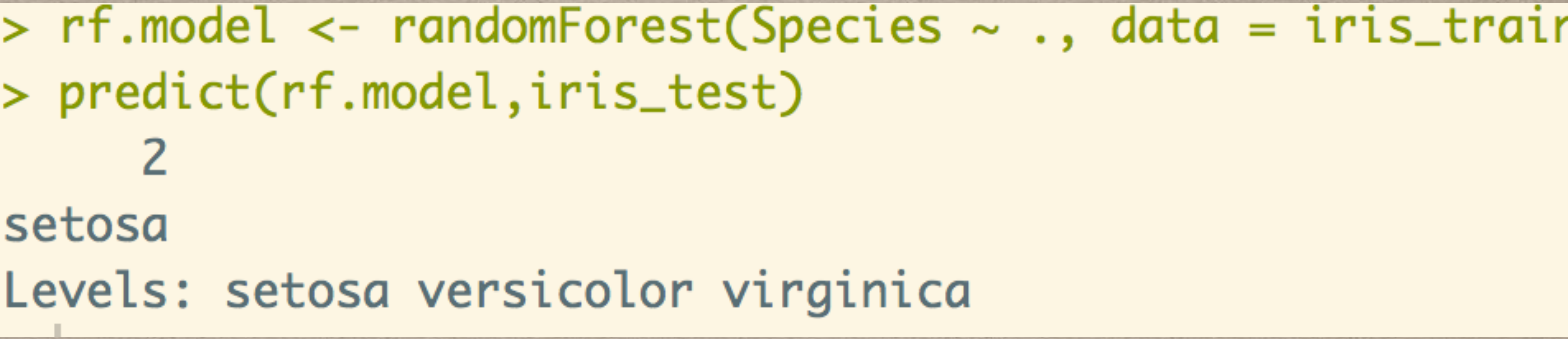
- Steps 1 to 4 is to build one tree.
- You repeat step 1 to 4 a large number of times to build a forest.
- There's no magic number for large number. You can build 101, 201, 501, 1001, etc.. There's research paper that suggest certain numbers but it base on your data. So just check model performance via model evaluation using cross validation.
- I have no idea why suggested numbers are usually even, but I choose odd number of trees in case of ties.

4. HOW TO USE RANDOM FOREST TO PREDICT

- You have an observation that you want to predict
- Say the observation is x = male, z = 23 year old
- You plug that in your your random forest.
- Random Forest takes those predictors and give it to each decision trees.
- Each decision trees give one prediction, cancer or no cancer (0,1).
- You take all of those predictions (aka votes) and take the majority.
- This is why I suggest odd number of trees to break ties for binary responses.

5. R CODE

```
set.seed(415)

library(randomForest)

iris_train <- iris[-1,]

iris_test <- iris[1,]

rf.model <- randomForest(Species ~ ., data = iris_train)

predict(rf.model,iris_test)
```



```
> rf.model <- randomForest(Species ~ ., data = iris_trai
> predict(rf.model,iris_test)
     2
setosa
Levels: setosa versicolor virginica
```

REFERENCES

- Statistical Learning from a Regression Perspective by Richard A. Berk
- Introduction to Data Mining by Pang-Ning Tan (http://www-users.cs.umn.edu/~kumar/dmbook/index.php)
- 11.12 From bagging to forest (https://onlinecourses.science.psu.edu/stat857/node/181)
- "An Empirical Comparison of Supervised Learning Algorithms" - https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf
- Random Forest created and trademarked by Leo Breiman (https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings)
- "Bagging and Random Forest Ensemble Algorithms for Machine Learning" By Jason Brownlee (http://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/)