# Twenty Questions

From Wikipedia, the free encyclopedia

*This article is about the spoken game. For the toy, see 20Q. For the computer-human game show, see 20Q (game show).*

**Twenty Questions** is a spoken parlor game, which encourages deductive reasoning and creativity. It originated in the United States and was played widely in the 19th century.[1] It escalated in popularity during the late 1940s, when it became the format for a successful weekly radio quiz program.

Opening titles on the *20 Questions* television panel show (1949–1955)

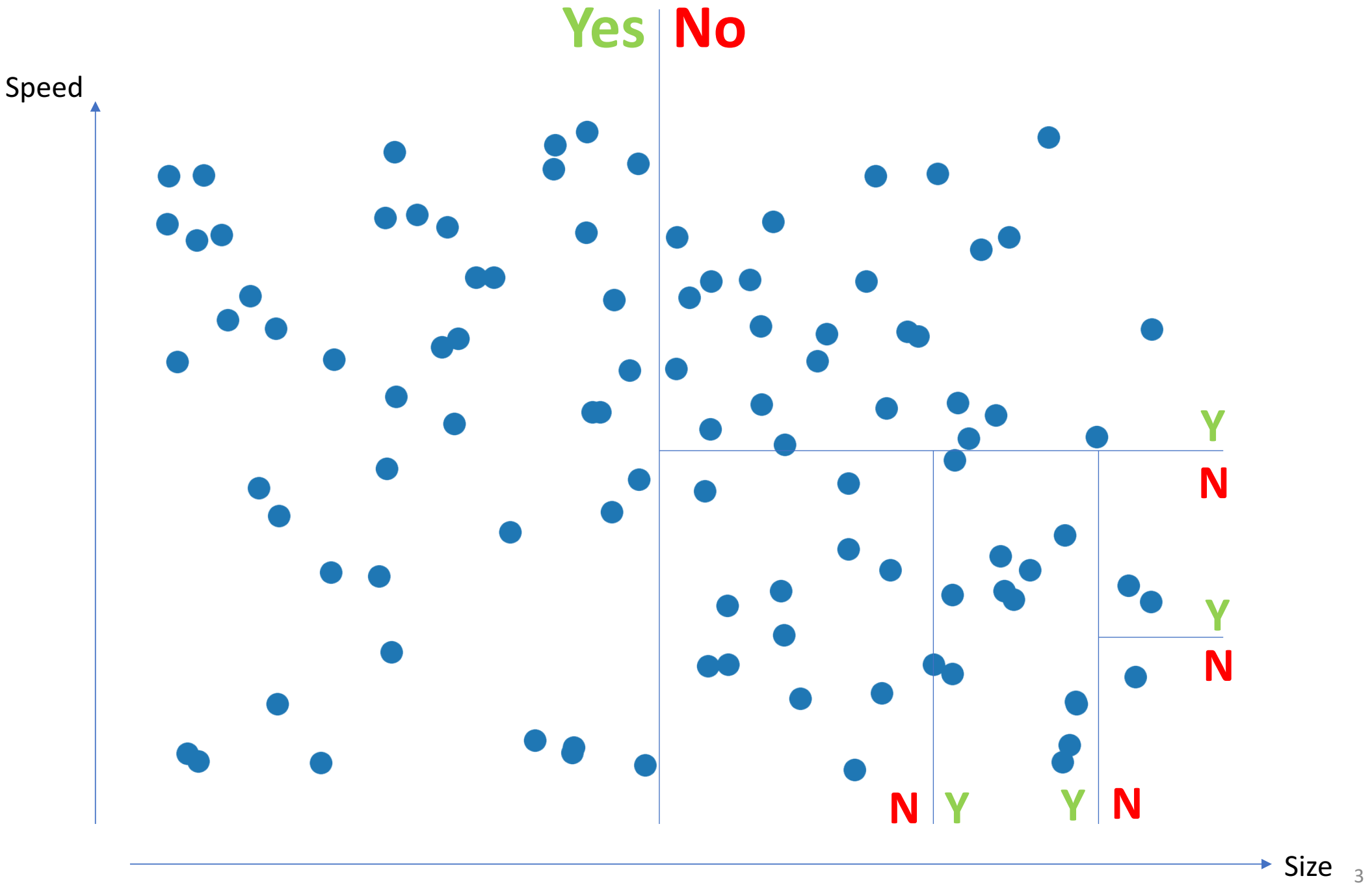
In the traditional game, one player is chosen to be the *answerer*. That person chooses a subject (object) but does not reveal this to the others. All other players are *questioners*. They each take turns asking a question which can be answered with a simple "Yes" or "No". In variants of the game, multiple state answers may be included such as the answer "Maybe". The answerer answers each question in turn. Sample questions could be: "Is it bigger than a breadbox?" or "Can I put it in my mouth?" Lying is not allowed in the game. If a questioner guesses the correct answer, that questioner wins and becomes the *answerer* for the next round. If 21 questions are asked without a correct guess, then the answerer has stumped the questioners and gets to be

slido

Join at

# slido.com

# #W255


Yes
No
Speed
Y
N
Y
N
N
Y
Y
N
Size

Answerer

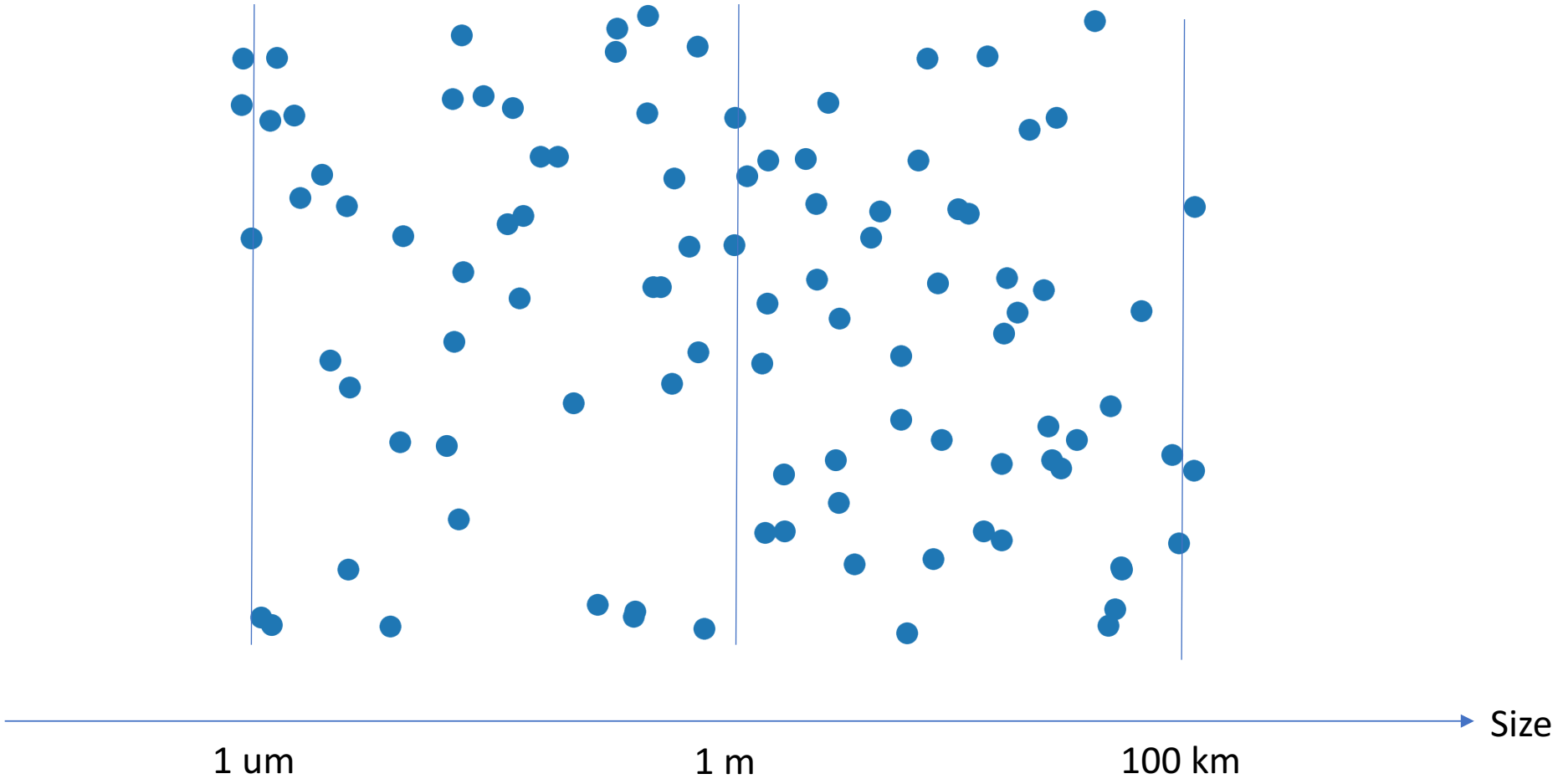
Pick something rare

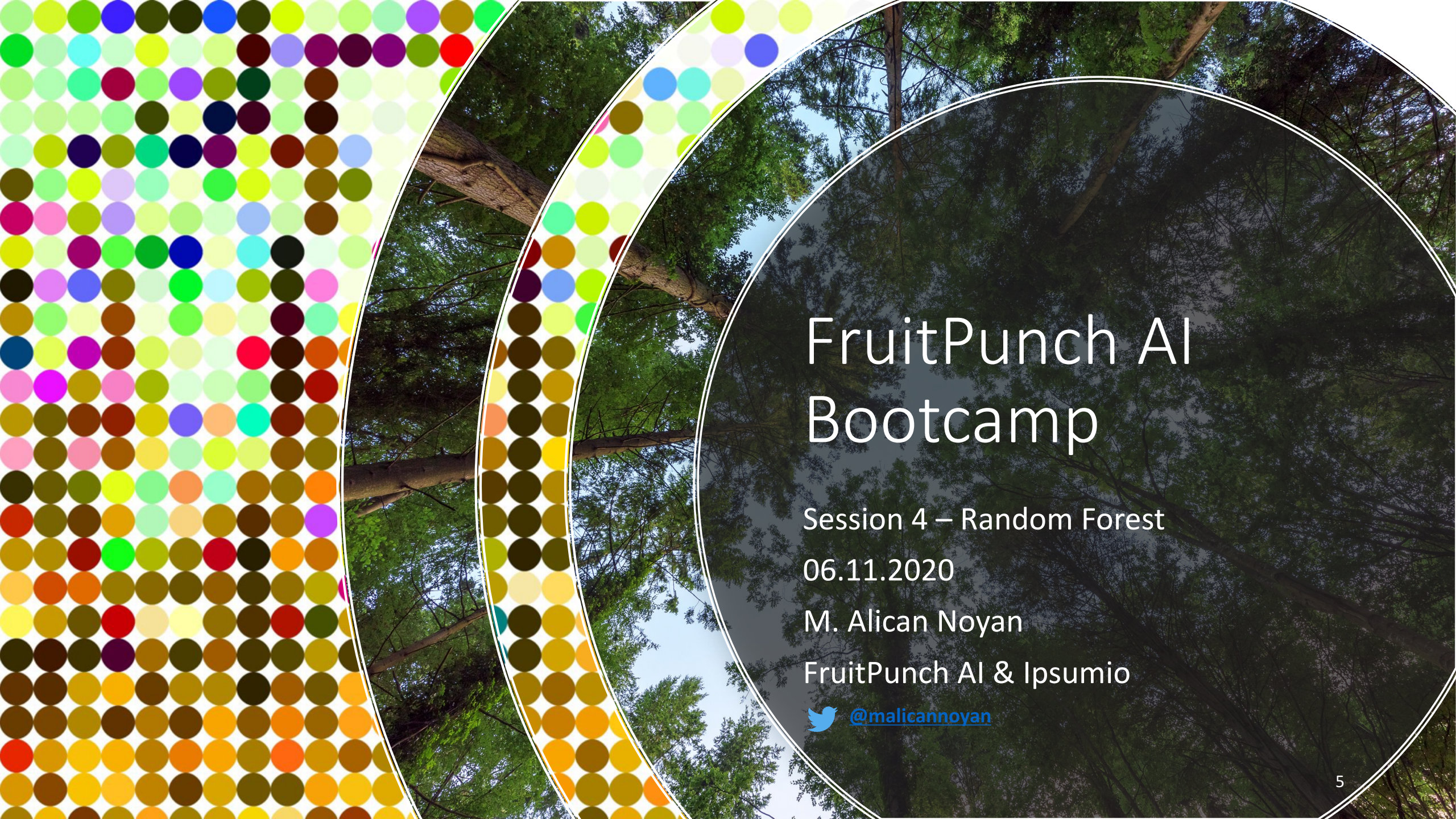
# What is the best strategy?

Questioner

Ask good questions

**What are good/bad questions?**





# FruitPunch AI Bootcamp

Session 4 – Random Forest

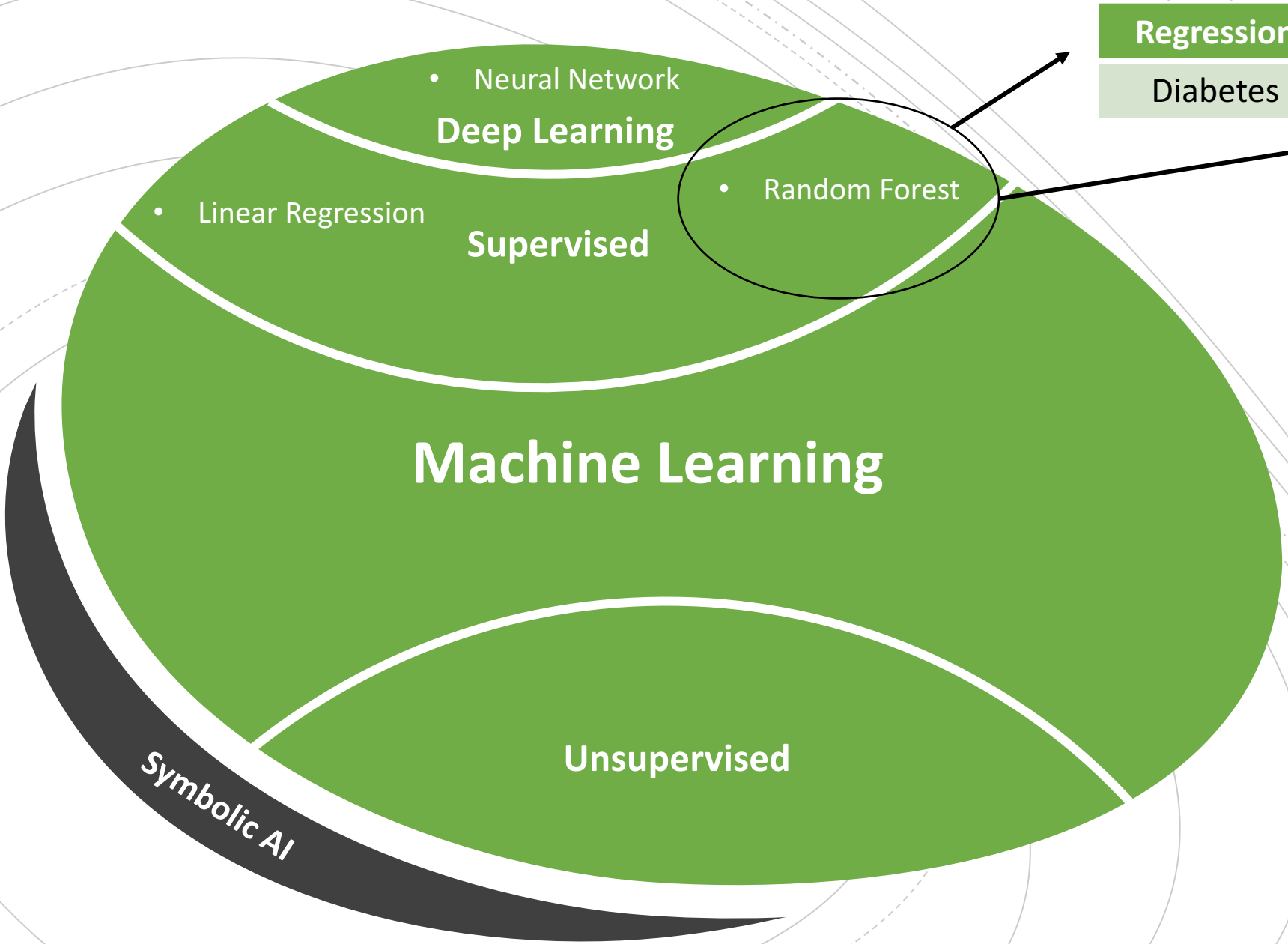
06.11.2020

M. Alican Noyan

FruitPunch AI & Ipsumio

@malicannoyan

# Artificial Intelligence



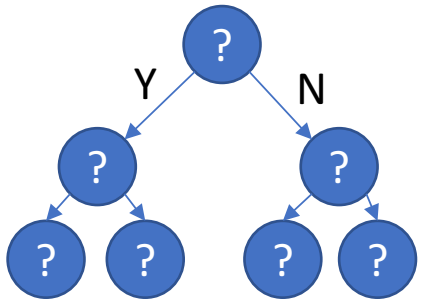

| Regression | Classification |
|---|---|
| Diabetes | Digits |

Unstructured
Neural Networks

**Simple** yet **powerful**

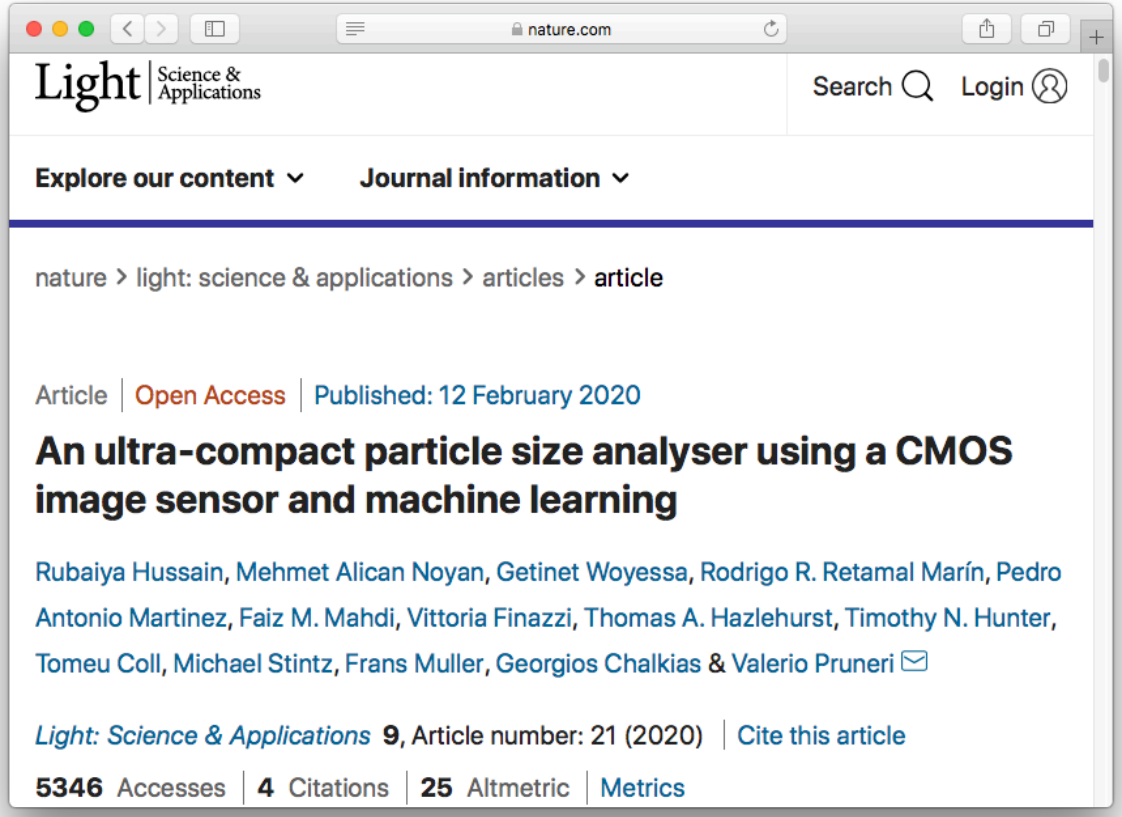
Structured
Random forest

**Decision Tree**




**Random Forest**

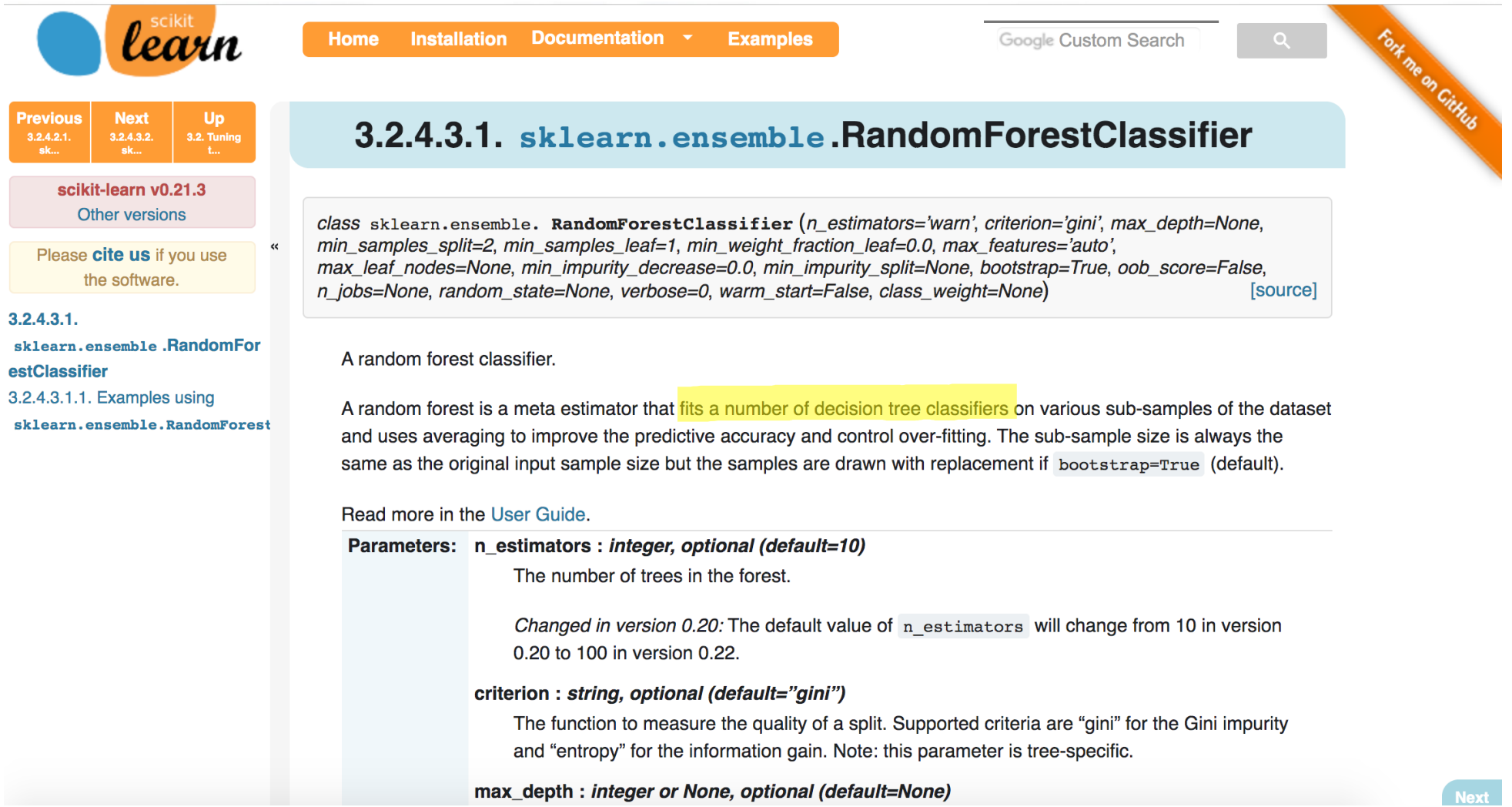
**Gradient Boosting**




**Today's challenge: Implement random forest algorithm for binary classification**

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X, y)
```



scikit learn

Home Installation Documentation Examples

Google Custom Search

Fork me on GitHub

Previous 3.2.4.2.1. sk... | Next 3.2.4.3.2. sk... | Up 3.2. Tuning t...

scikit-learn v0.21.3
Other versions

Please cite us if you use the software.

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier
3.2.4.3.1.1. Examples using sklearn.ensemble.RandomForest

# 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

*class* sklearn.ensemble. **RandomForestClassifier** (*n_estimators='warn', criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the User Guide.

**Parameters:**

**n_estimators : *integer, optional (default=10)***

The number of trees in the forest.

*Changed in version 0.20:* The default value of `n_estimators` will change from 10 in version 0.20 to 100 in version 0.22.

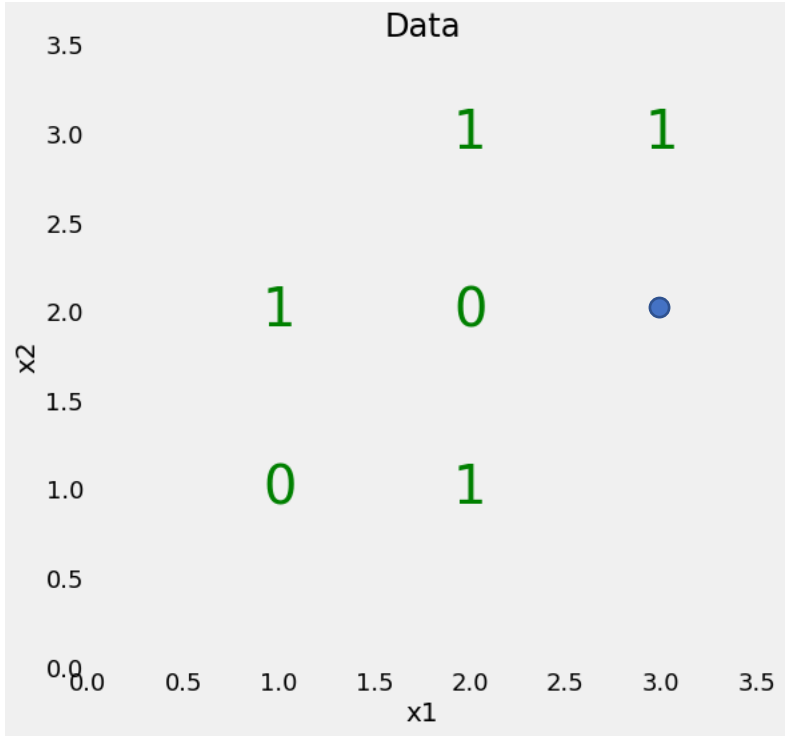
**criterion : *string, optional (default="gini")***

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

**max_depth : *integer or None, optional (default=None)***

Next

# Decision tree




x1 = 3, x2 = 2

x2 <= 2.5
gini = 0.444
samples = 6
value = [2, 4]
class = 1

True

False

(3, 2)

x2 <= 1.5
gini = 0.5
samples = 4
value = [2, 2]
class = 0

gini = 0.0
samples = 2
value = [0, 2]
class = 1

x1 <= 1.5
gini = 0.5
samples = 2
value = [1, 1]
class = 0

x1 <= 1.5
gini = 0.5
samples = 2
value = [1, 1]
class = 0

(3, 2)

gini = 0.0
samples = 1
value = [1, 0]
class = 0

gini = 0.0
samples = 1
value = [0, 1]
class = 1

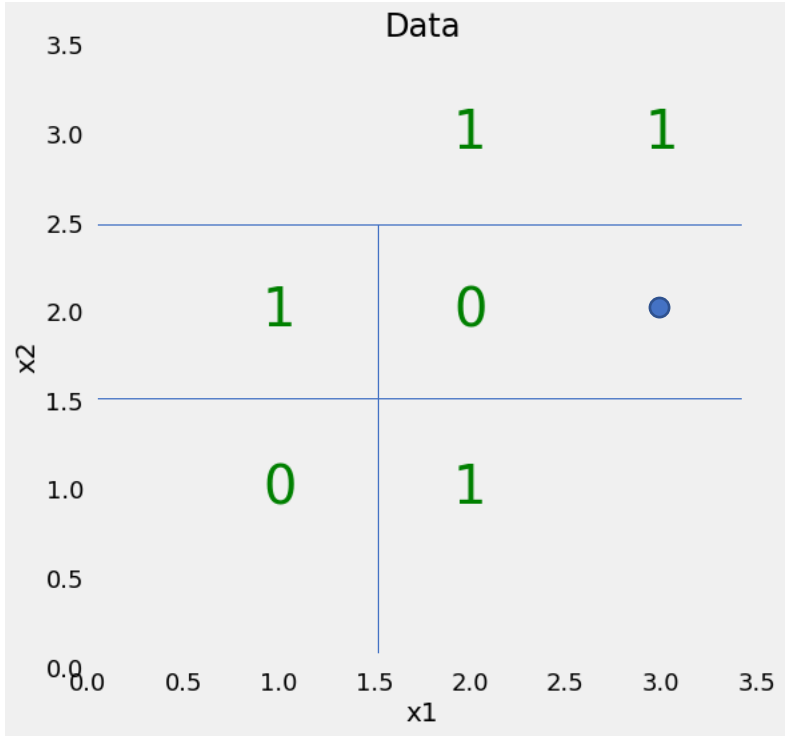
gini = 0.0
samples = 1
value = [0, 1]
class = 1

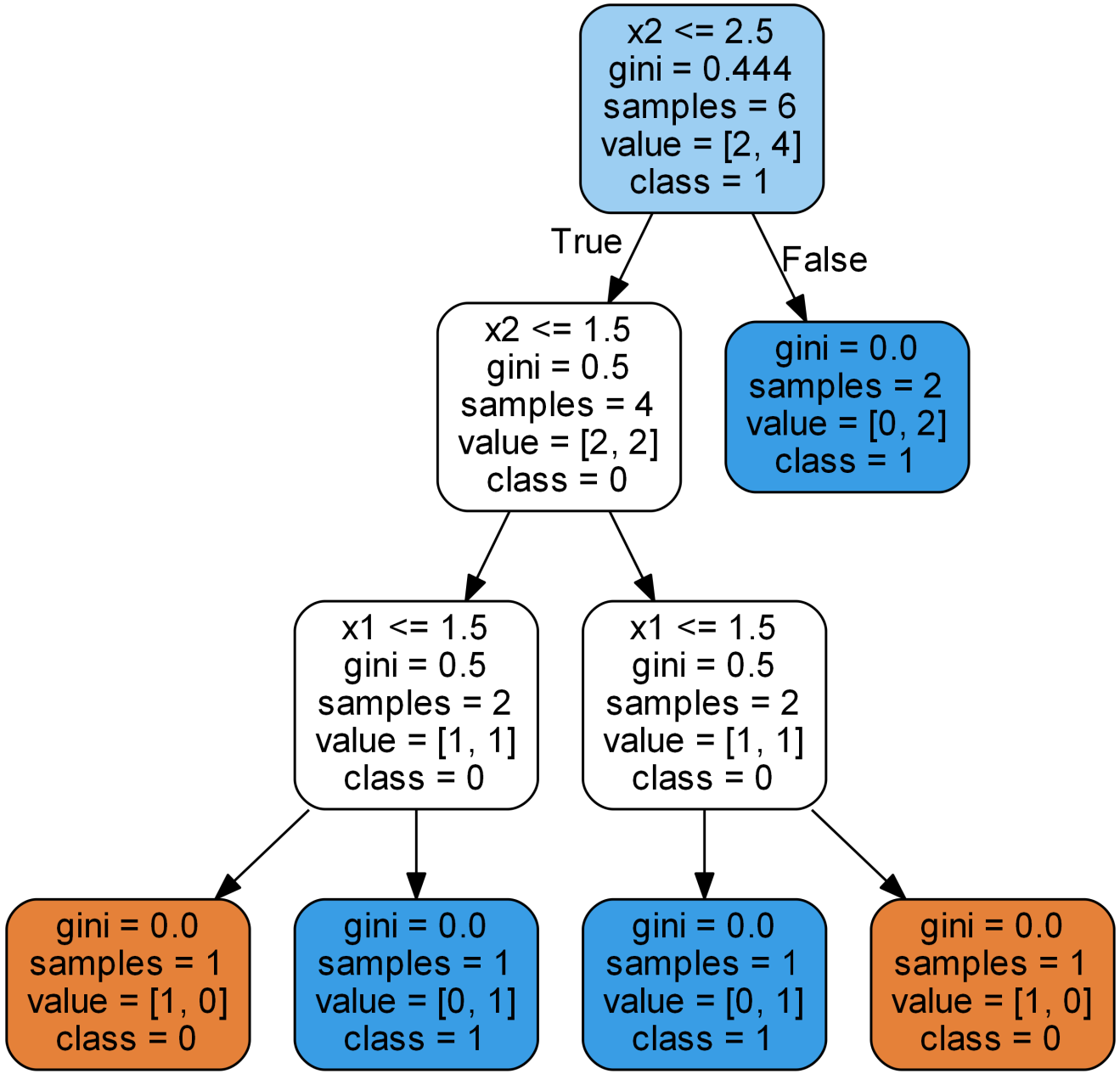
gini = 0.0
samples = 1
value = [1, 0]
class = 0

(3, 2)

Will Koehresen https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

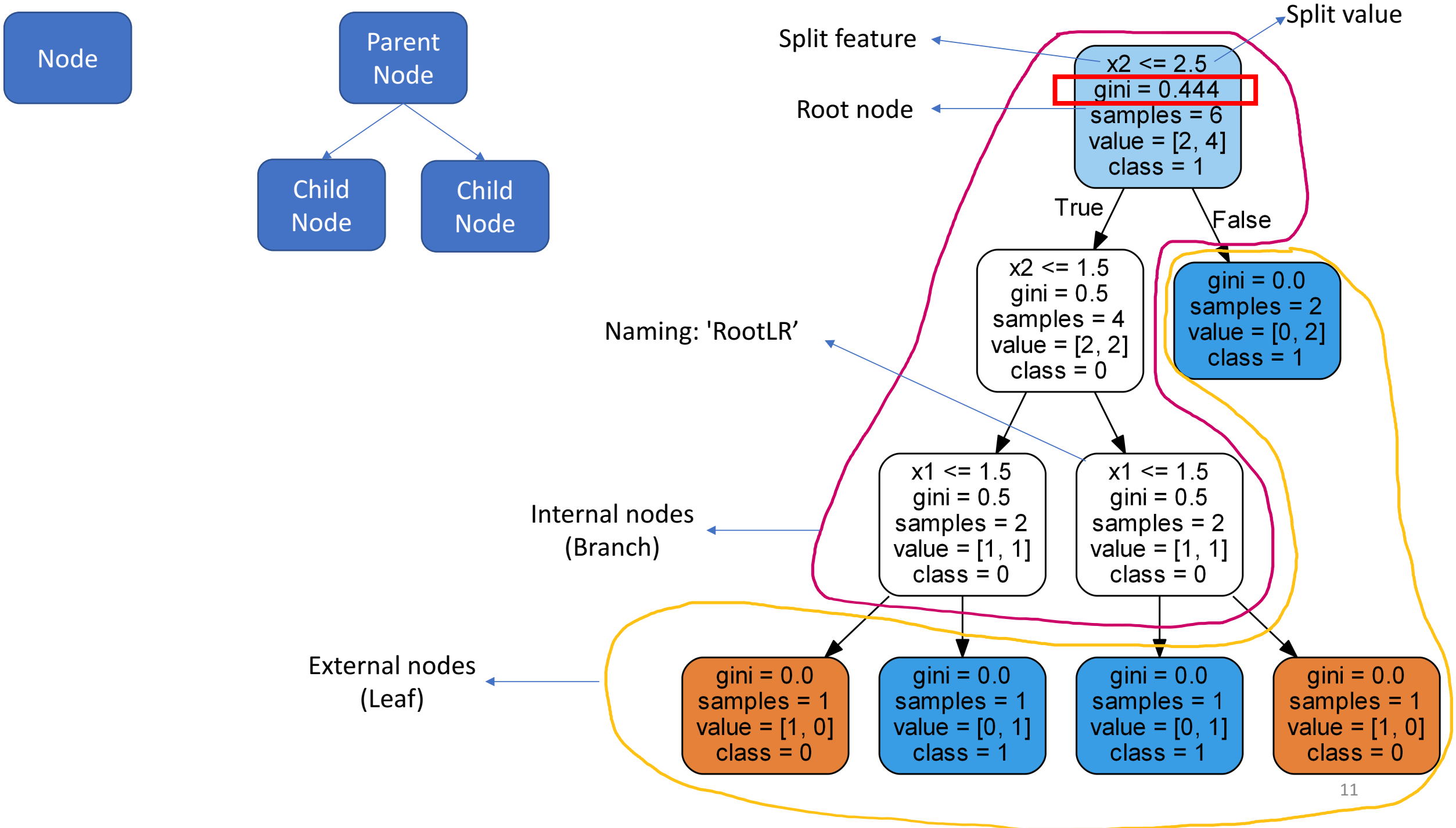
# Decision tree



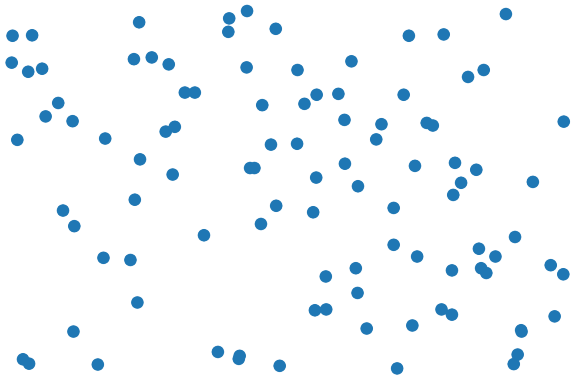





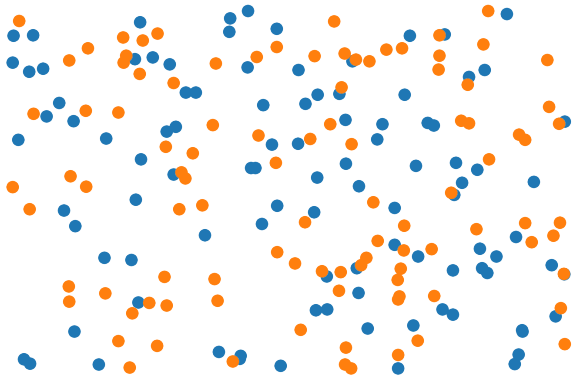

Will Koehresen https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76


Node
Parent Node
Child Node
Child Node
Split feature
Split value
Root node
x2 <= 2.5
gini = 0.444
samples = 6
value = [2, 4]
class = 1
True
False
x2 <= 1.5
gini = 0.5
samples = 4
value = [2, 2]
class = 0
gini = 0.0
samples = 2
value = [0, 2]
class = 1
Naming: 'RootLR'
Internal nodes
(Branch)
x1 <= 1.5
gini = 0.5
samples = 2
value = [1, 1]
class = 0
x1 <= 1.5
gini = 0.5
samples = 2
value = [1, 1]
class = 0
External nodes
(Leaf)
gini = 0.0
samples = 1
value = [1, 0]
class = 0
gini = 0.0
samples = 1
value = [0, 1]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 1
gini = 0.0
samples = 1
value = [1, 0]
class = 0

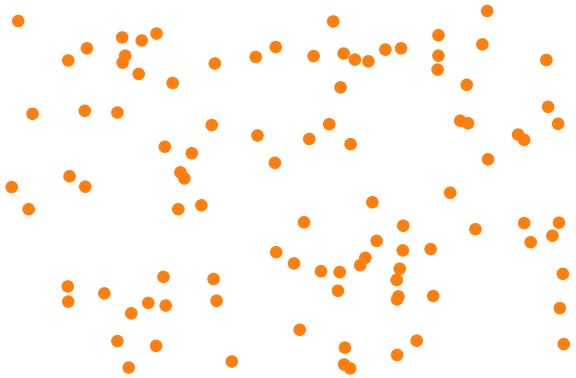
# Gini impurity



0



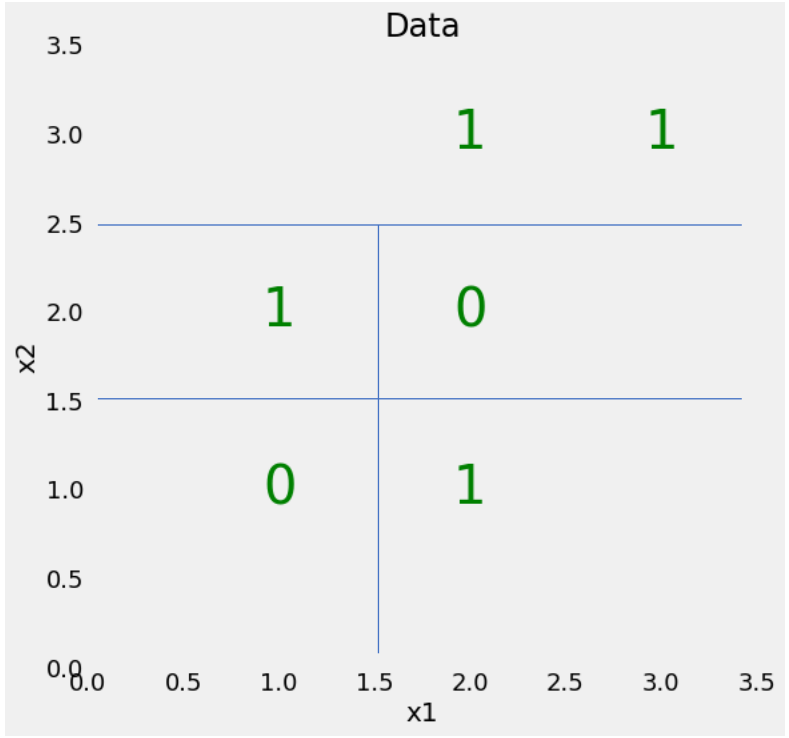
0.5

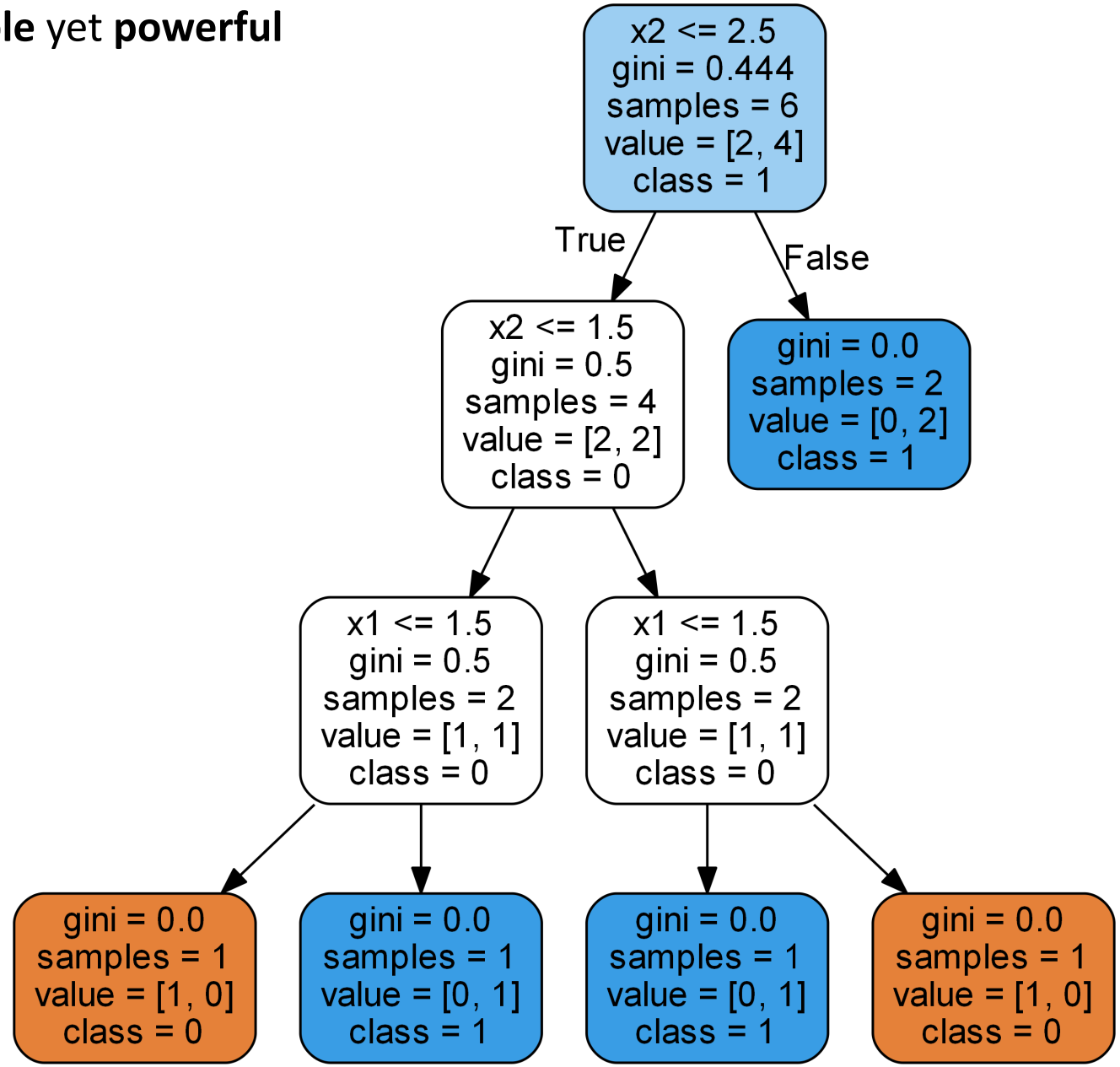


0

# Decision tree

**Simple** yet **powerful**






Will Koehresen https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

## Gini impurity [ edit ]

*Not to be confused with Gini coefficient.*

Used by the CART (classification and regression tree) algorithm for classification trees, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The Gini impurity can be computed by summing the probability $p_i$ of an item with label $i$ being chosen times the probability $\sum_{k \neq i} p_k = 1 - p_i$ of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

To compute Gini impurity for a set of items with $J$ classes, suppose $i \in \{1, 2, \ldots, J\}$, and let $p_i$ be the fraction of items labeled with class $i$ in the set.

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = \boxed{1 - \sum_{i=1}^{J} {p_i}^2}$$

$$= 1 - p_0^2 - p_1^2$$

[1, 1, 1, 1, 0, 0]

$= 1 - (4/6)^2 - (2/6)^2$

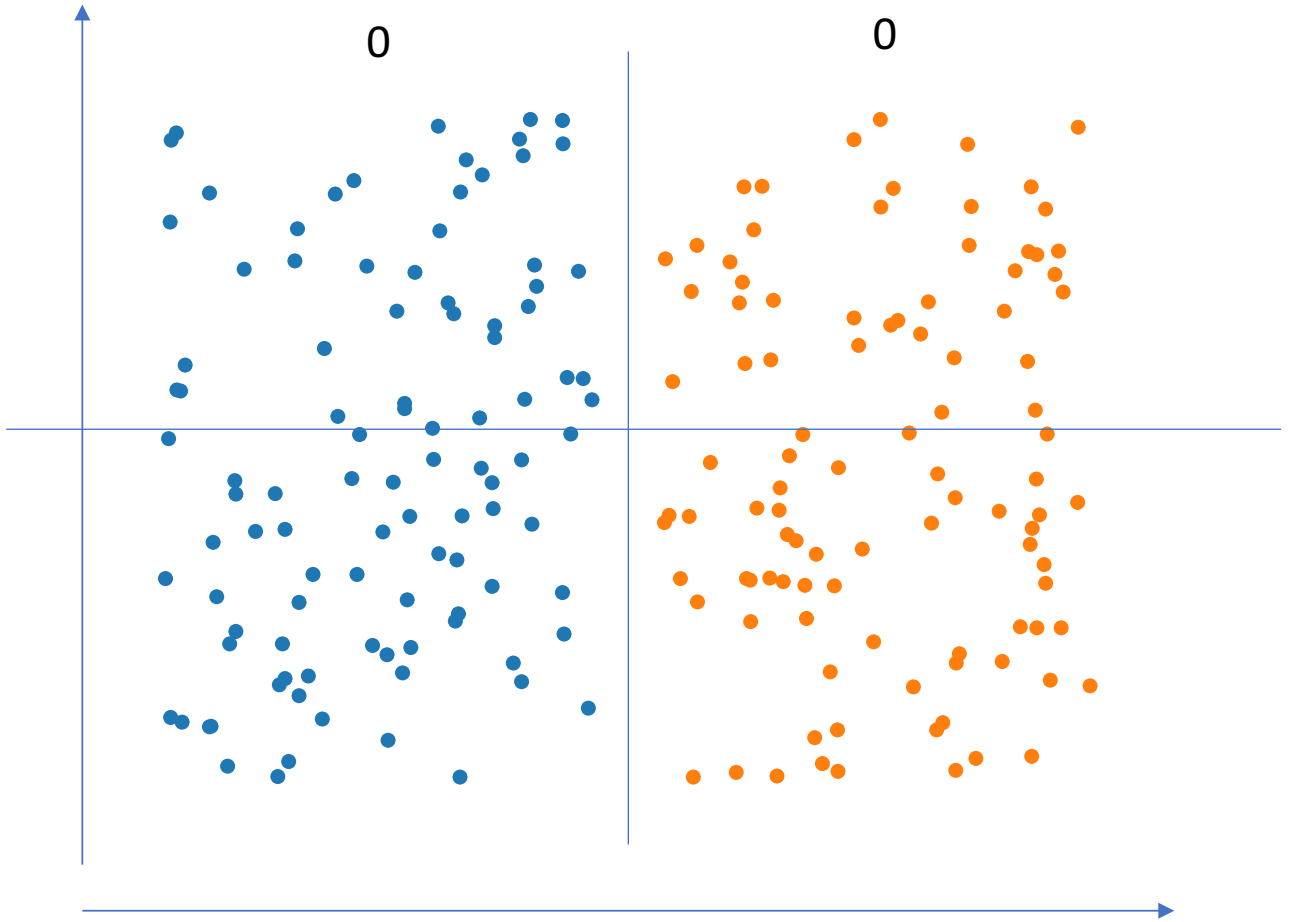
$= 0.44$

x2 <= 2.5
gini = 0.444
samples = 6
value = [2, 4]
class = 1

**Challenge 1: Implement a gini calculator**

# How find the split using gini?

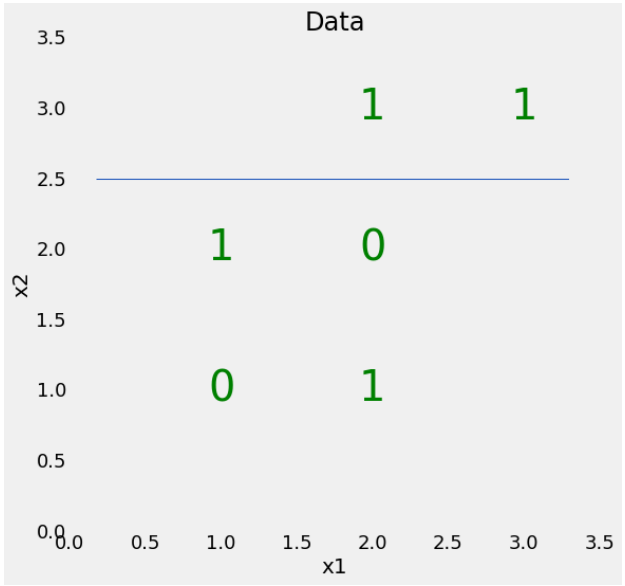
Weighted average gini = 0



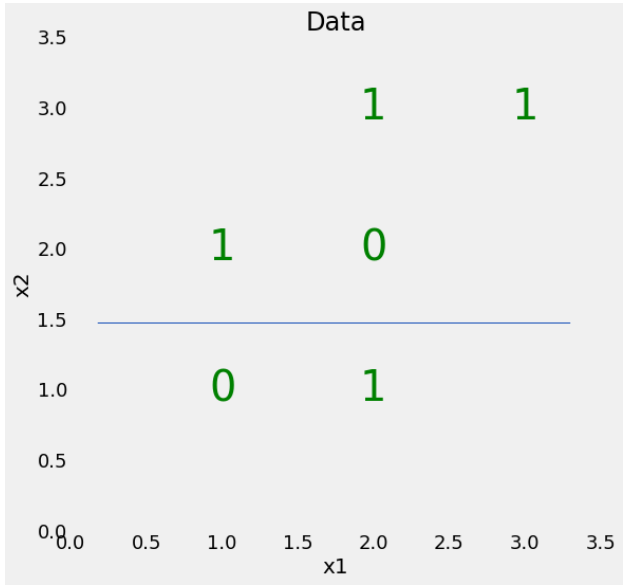

Weighted average gini = 0.5

**Gini of a split: weighted avg of splitted subsets (Implemented)**

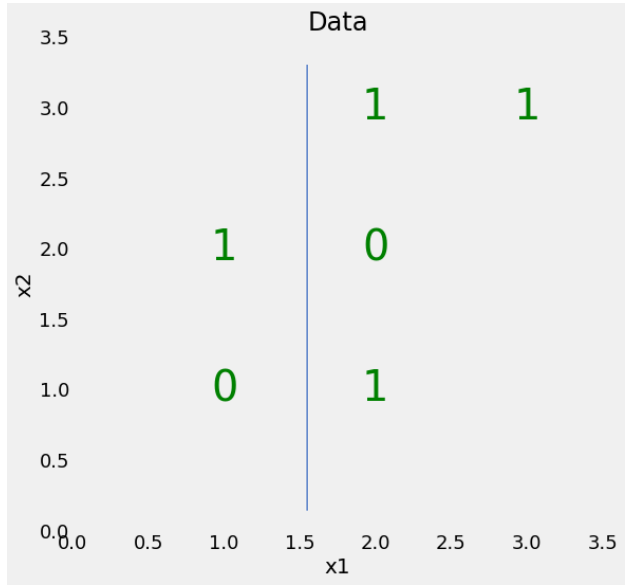
# Find the best split



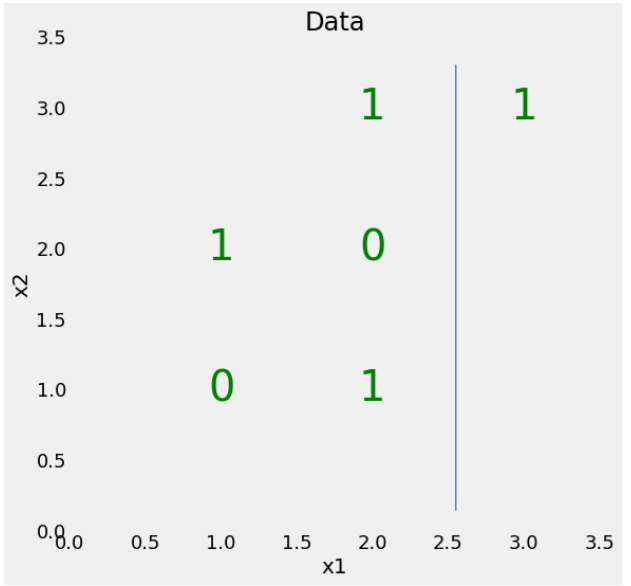

Avg gini 0.3333




0.4166

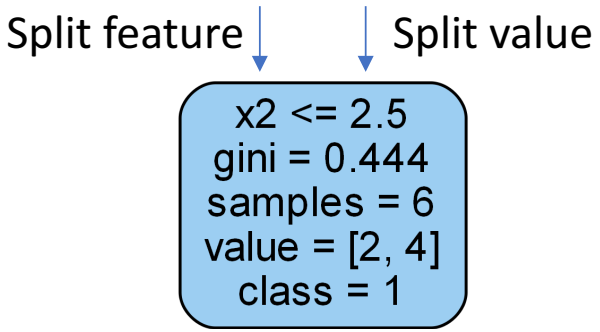



0.4166

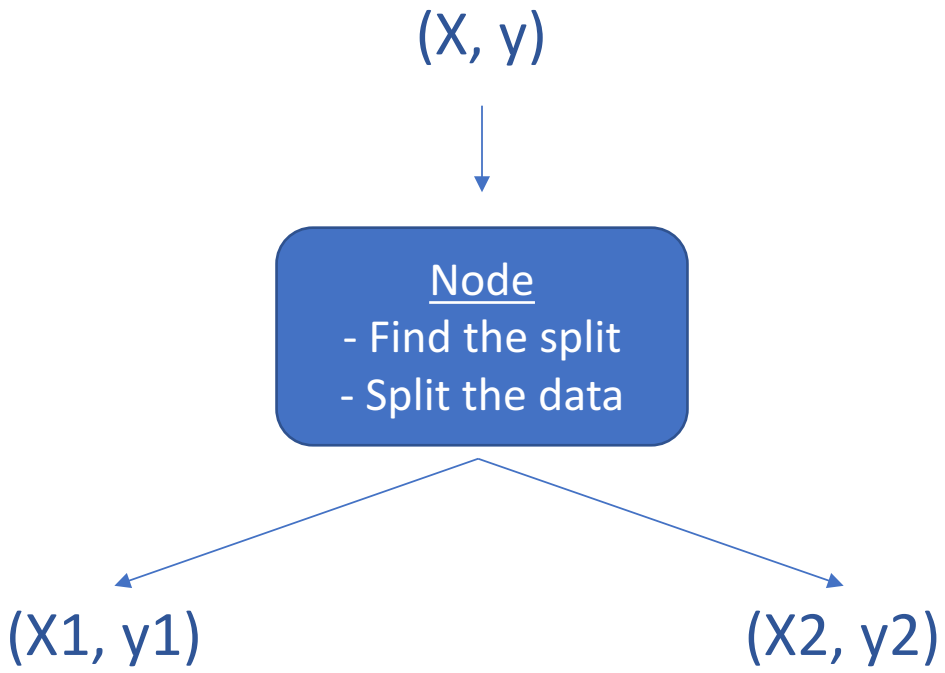



0.4

## Challenge 2: Find the best split

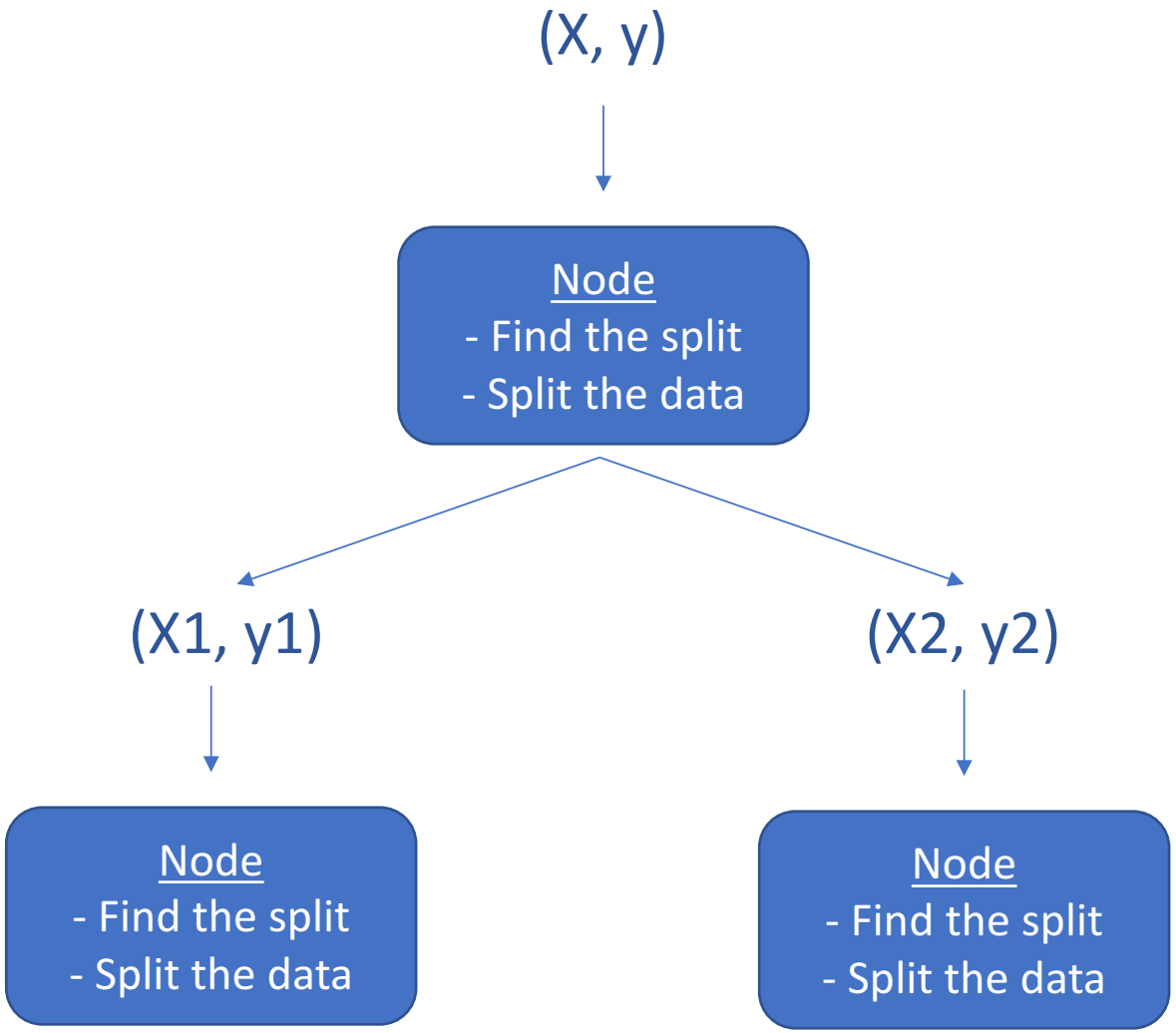


# A node




**Splitter: Implemented**

# Challenge 3 – Implement fitting a tree




Termination conditions:

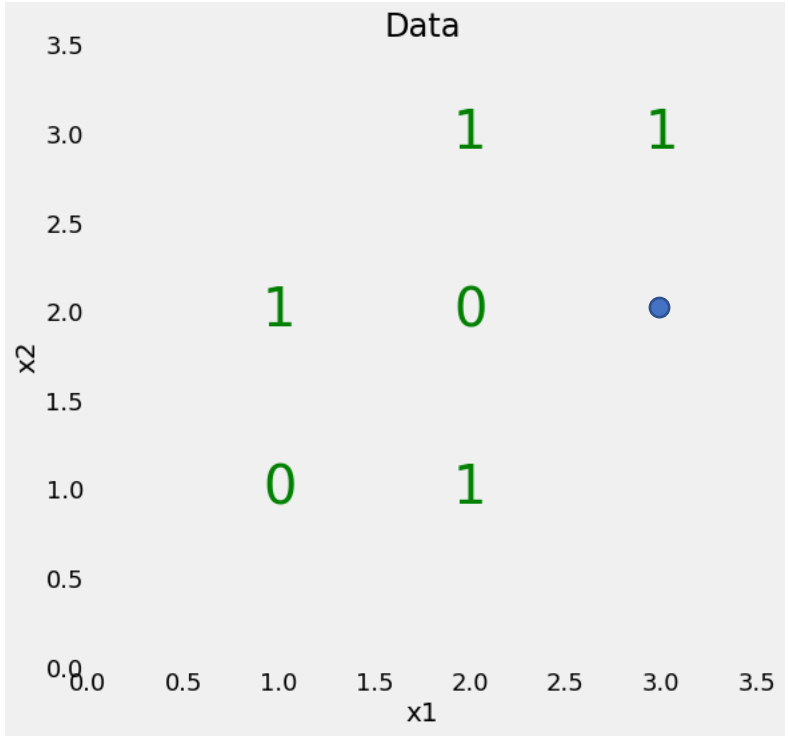
Gini = 0

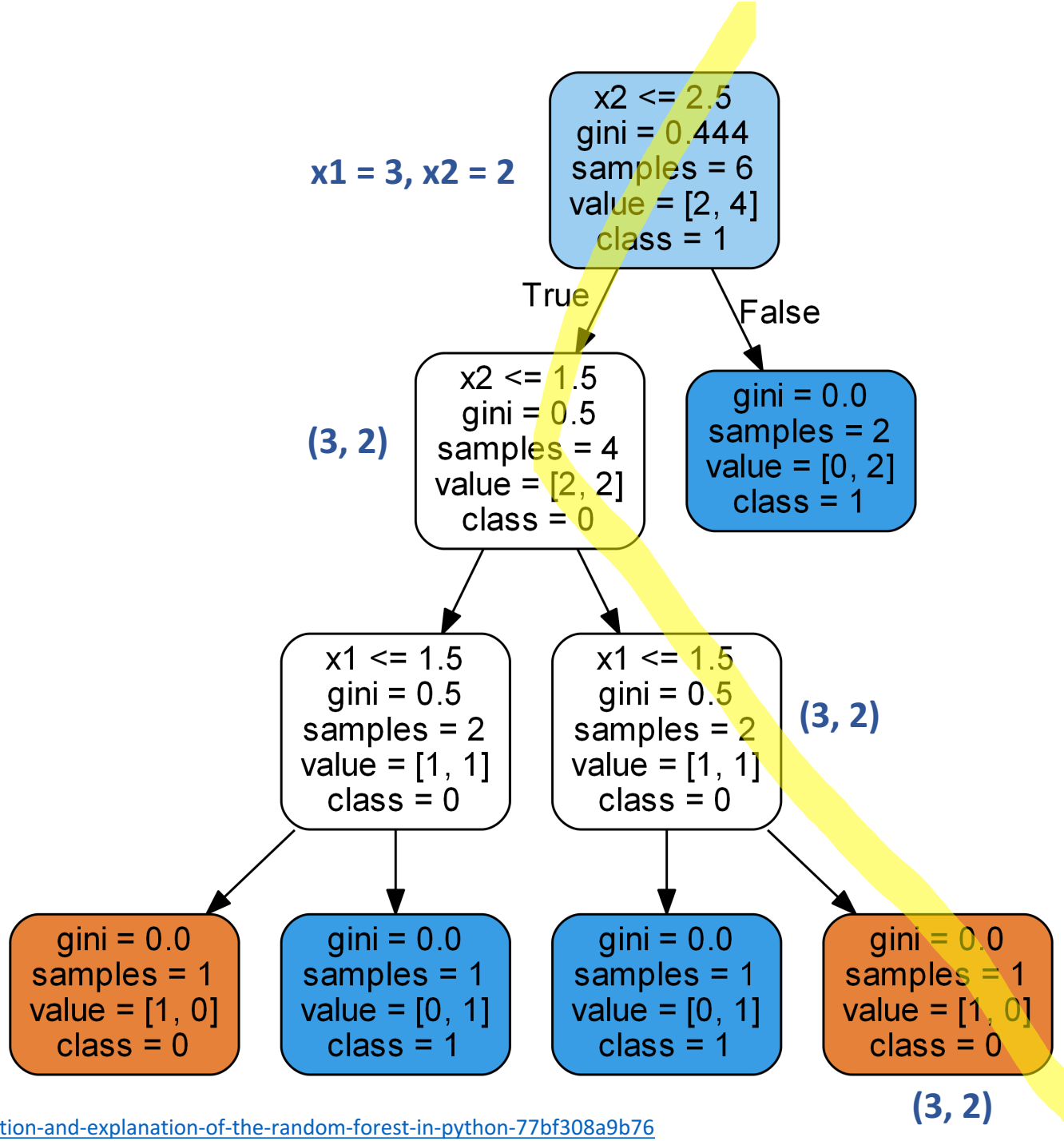
Limit depth

Limit min data at leaf

Return tree for predictions

# Challenge 4 – Implement prediction




Will Koehresen https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

- Gini calculator (C1)
- Gini of a split (Done)
- Split finder (C2)
- Node – Splitter (Done)
- Repeat nodes to fit the tree (C3)
- Predict using trained tree (C4)

# Random Forest

- Randomly sample from X and y and fit 30 trees (Challenge 5)
- Predict using 30 trees and output the average (Challenge 6)