TÉCNICO LISBOA

Planning, Learning and Decision Making

Lecture 12. Learning from examples - Decision trees

What is learning?

Can computers improve their performance with experience/data?

What is ML?

- “Field of study that gives computers the ability to learn without being explicitly programmed”

Arthur Samuel, 1901-1990

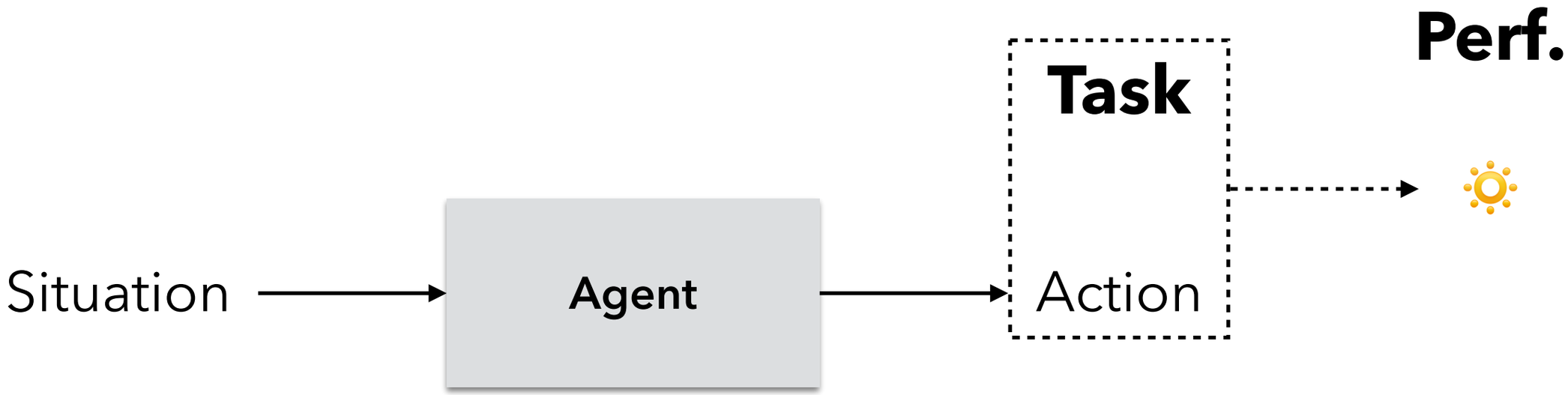
Creator of world’s first self-learning program

What is ML?

- "A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."

Tom Mitchell, 1951-today

Author of book "Machine Learning"


Perf.
Task
Situation
Agent
Action

Experience

Situation → **Agent** → Action

Task

Perf.

Learning problem

- Specifying a learning problem involves:
  - Identifying the **task**, **T**
  - Defining the **performance measure**, **P**
  - Identifying/selecting the **training experience**, **E**

Example

- **Handwriting recognition**
  - **Task:** Recognize and classify handwritten words from images
  - **Performance:** Percent of words correctly classified
  - **Training experience:** Database of handwritten words and corresponding classifications

Example

- **Automated driving**
  - **Task:** Drive in a public highway using vision
  - **Performance:** Average distance travelled before error (detected by human)
  - **Training experience:** Sequence of images and steering commands recorded from human driver

Example

- **Checkers**
  - **Task:** Play checkers competently
  - **Performance:** Percentage of games won against different opponents
  - **Training experience:** Playing practice games against itself

Types of learning

Types of learning I

- **Learning from examples**
  - **Task:** Learning a situation-action correspondence (decision rule)
  - **Performance:** Correctness of actions in all possible situations
  - **Training experience:** Set of situation-action pairs

Supervised learning

Example

- **Spam filter**
  - Input: Specific e-mail message
  - Output: Decision of wether mail is spam or not
  - Filter is “trained” from set of examples of spam and non-spam

Discrete output

Example

- **Spam filter**
  - Input: Specific e-mail message
  - Output: Decision of wether mail is spam or not
  - Filter is “trained” from set of examples of spam and non-spam

Binary classification

Example

- **Handwritten digit recognition**
  - Input: Image
  - Output: Decision of which digit it is
  - System is “trained” from set of examples of digit images

Discrete output

Example

- **Handwritten digit recognition**
  - Input: Image
  - Output: Decision of which digit it is
  - System is “trained” from set of examples of digit images

Classification (general)

Example

- **Automated driving**
  - Input: Sequence of images
  - Output: Decision regarding steering angle
  - System is “trained” from set of examples of image-steering commands

Continuous output

Example

- **Automated driving**
  - Input: Sequence of images
  - Output: Decision regarding steering angle
  - System is “trained” from set of examples of image-steering commands

Regression

Types of learning II

- **Learning by trial-and-error**
  - **Task:** Learning a situation-action correspondence (decision rule)
  - **Performance:** Accumulated cost of actions in all possible situations
  - **Training experience:** Set of situation-action-cost triplets

Reinforcement learning

Example

- **Checkers player**
  - Input: Board description
  - Output: Decision of which play to make next
  - System trains by playing games against itself

Example

- **Robot motion**
  - Input: Robot pose
  - Output: Decision of which motion to execute next
- System trains by exploring space

Types of learning III

- **Find structure in data**
  - **Task:** Find hidden structure in data
  - **Performance:** Quality of structure found
  - **Training experience:** Data to be analyzed

- We will not be doing this

Unsupervised learning

Learning from examples

Learning from examples

- We provide the agent with **examples** of situations-actions
- We would like the agent to learn the **underlying task**
  - Perform similarly in known situations
  - Generalize to situations never encountered before

Focus on situations with 2 actions
(other are similar)

How to learn?

Example

- Compute a binary function of state x
- State is described by 3 *features*, $\phi_1(x)$, $\phi_2(x)$, $\phi_3(x)$
- Examples:

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	a
0	1	0	0
0	0	1	0
0	1	1	1
1	0	0	1

What is the function?

Two solutions

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	a
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	a
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

What is the problem?

No free lunches…

- Beyond the given examples, everything is possible!

No free lunch theorem in machine learning

For every task in which your decision rule does great, there is another task in which it does terrible…

So is there no hope?

No free lunches?

- Some hypothesis are **much more likely** than others
- We build our algorithms around **assumptions** on the problem
- Assumptions translate knowledge about the problem (which hypotheses are most likely)
- Assumptions greatly reduce search space (this reduction is known as **inductive bias**)

Example

- Compute a binary function of $\phi_1(x)$, $\phi_2(x)$ and $\phi_3(x)$, involving only conjunctions and disjunctions
- Hypotheses:
  - $\phi_1(x) \vee \phi_2(x) \vee \phi_3(x)$
  - $(\phi_1(x) \wedge \phi_2(x)) \vee \phi_3(x)$
  - $\phi_1(x) \wedge (\phi_2(x) \vee \phi_3(x))$
  - $\phi_1(x) \vee (\phi_2(x) \wedge \phi_3(x))$
  - $(\phi_1(x) \vee \phi_2(x)) \wedge \phi_3(x)$
  - $\phi_1(x) \wedge \phi_2(x) \wedge \phi_3(x)$

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	a
0	1	0	0
0	0	1	0
0	1	1	1
1	0	0	1

What is the function?

Example

- Compute a binary function of $\phi_1(x)$, $\phi_2(x)$ and $\phi_3(x)$, involving only conjunctions and disjunctions
- Hypotheses:
  - $\phi_1(x) \vee \phi_2(x) \vee \phi_3(x)$
  - $(\phi_1(x) \wedge \phi_2(x)) \vee \phi_3(x)$
  - $\phi_1(x) \wedge (\phi_2(x) \vee \phi_3(x))$
  - $\phi_1(x) \vee (\phi_2(x) \wedge \phi_3(x))$
  - $(\phi_1(x) \vee \phi_2(x)) \wedge \phi_3(x)$
  - $\phi_1(x) \wedge \phi_2(x) \wedge \phi_3(x)$

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	a
0	1	0	0
0	0	1	0
0	1	1	1
1	0	0	1

$$\phi_1(x) \vee (\phi_2(x) \wedge \phi_3(x))$$

Example

- More importantly: we can now decide in novel situations:
  - What is the output for the input (1, 1, 0)?
- R: 1

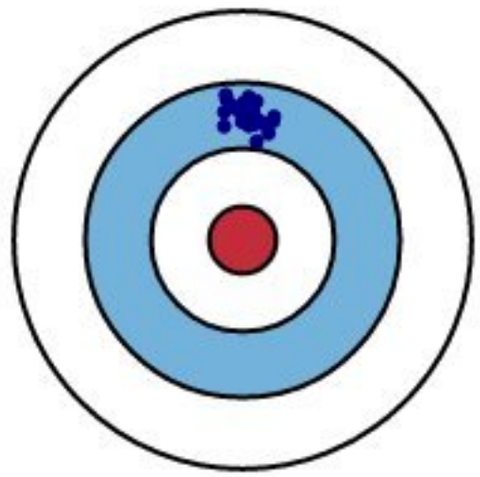
$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	a
0	1	0	0
0	0	1	0
0	1	1	1
1	0	0	1

Inductive bias

- Assumptions about the input-output relation allow us to **generalize**
  - Limit set of hypotheses under consideration
  - Reduce the set of hypotheses that match the observed data
  - Able to generalize the input-output relation in the observed data to unseen inputs

Inductive bias (cont.)

- **Wrong assumptions**
  - The hypotheses considered poorly represent the desired input-output correspondence
  - Learned hypothesis will not properly represent the data

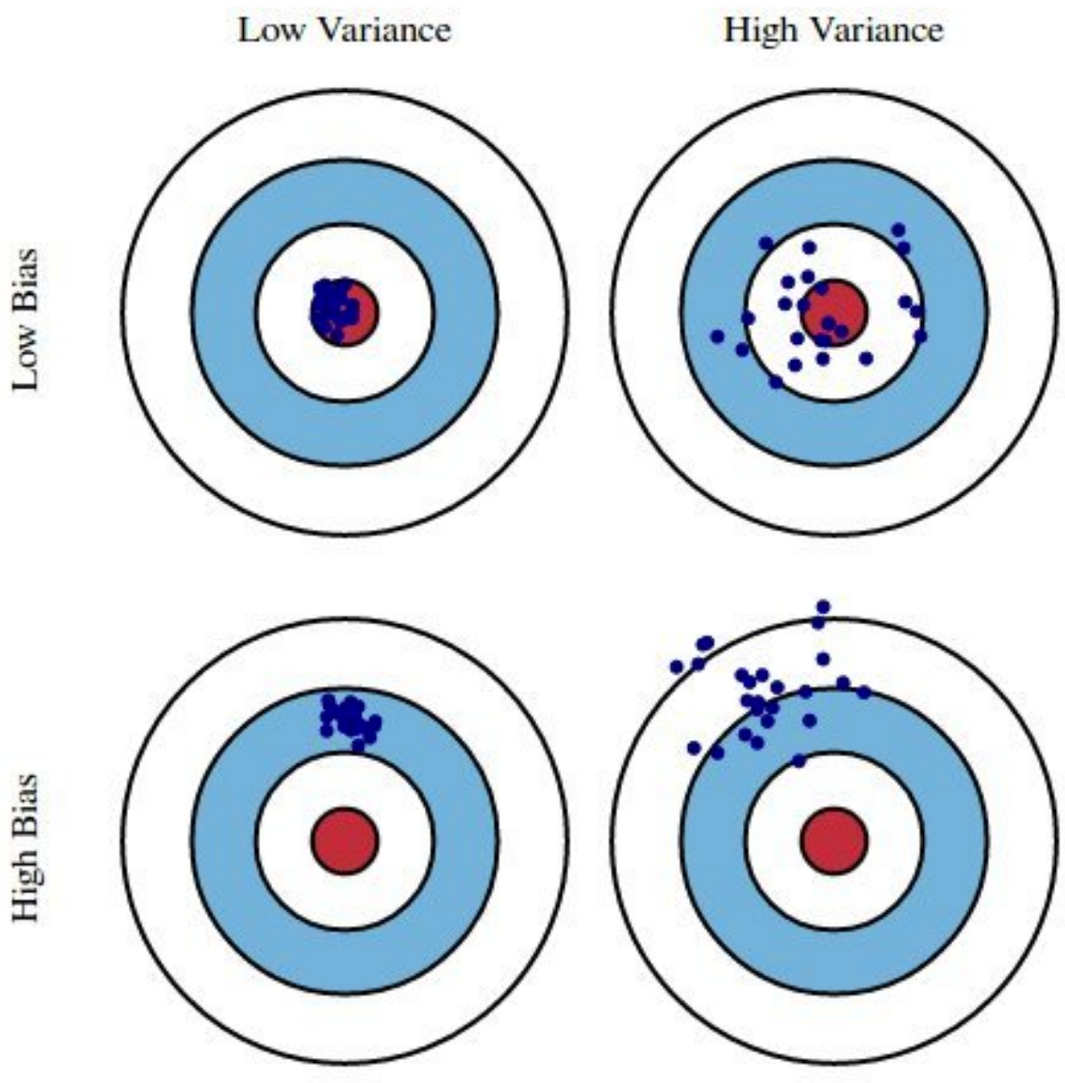


Tackling bias

- Make as few assumptions as possible?
  - Set of possible hypotheses “sufficiently large”
  - Make sure that the desired input-output correspondence is considered

Bias-variance trade-off

- Sources of error in learning algorithm:
  - **Bias:** error introduced by assumptions about the target hypothesis
  - **Variance:** error introduced due to the variability of the training set
- Assumptions must compromise between the two



The learning process

Some nomenclature

- Set of possible situations, $\mathcal{X}$ → Input space
- Set of possible actions, $\mathcal{A}$ → Output space

Some nomenclature

TASK:

- Decision rule (policy): ⟶ Classifier
  - Deterministic $\pi : \mathcal{X} \to \mathcal{A}$ ⟶ Discriminant function
  - Stochastic $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ ⟶ Discriminative model

Some nomenclature

EXPERIENCE:

- Dataset of examples $\mathscr{D} = \{(x_0, a_0), (x_1, a_1), \ldots, (x_M, a_M)\}$
- Pairs (x, a) generated from an unknown distribution μ_D

$$\mu_D(x, a) = \mathbb{P}\left[\mathrm{x} = x, \mathrm{a} = a\right]$$

Some nomenclature

PERFORMANCE:

- Expected cost or risk of policy π:

$$L(\pi) = \mathbb{E}_{\mu_D}\left[\ell(\mathrm{x}, \mathrm{a}; \pi)\right] = \sum_{x,a} \ell(x, a; \pi)\mu_D(x, a)$$

Input-output distribution

Cost of π given correspondence (*x*, *a*)

Some nomenclature

PERFORMANCE:

- Sometimes, Q can be computed from a simpler loss function:

$$\ell(x, a; \pi) = \mathbb{E}_{\pi}\left[\mathsf{d}(\hat{a}, a)\right] = \sum_{\hat{a}} \mathsf{d}(\hat{a}, a)\pi(\hat{a} \mid x)$$

- Goal of the agent is to compute a policy

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\, L(\pi)$$

Problem

- Given a policy π, we want to compute

$$L(\pi) = \mathbb{E}_{\mu_D}\left[\ell(\mathrm{x}, \mathrm{a}; \pi)\right] = \sum_{x,a} \ell(x, a; \pi)\mu_D(x, a)$$

Evaluate error in all situations?

Alternative 1

- Evaluate the policy in the provided examples:

$$L(\pi) \approx \hat{L}_N(\pi) = \frac{1}{N} \sum_{n=1}^{N} \ell(x_n, a_n; \pi)$$

Empirical risk

Empirical risk minimization

Alternative 2

- Estimate distribution $\mu_D(x, a)$ from the examples
- Select policy π*

$$\pi^* = \underset{\pi}{\operatorname{argmin}} \sum_{x,a} \hat{\mu}(x,a)\ell(x,a;\pi)$$

Inference on the data distribution

Alternative 2

- Estimate distribution $\mu_D(x, a)$ from the examples
- Select policy π^* (given simple loss d)

$$\pi^*(x) = \operatorname*{argmin}_{\hat{a}} \sum_{a} \hat{\mu}(x, a) \mathsf{d}(\hat{a}, a)$$

Inductive learning

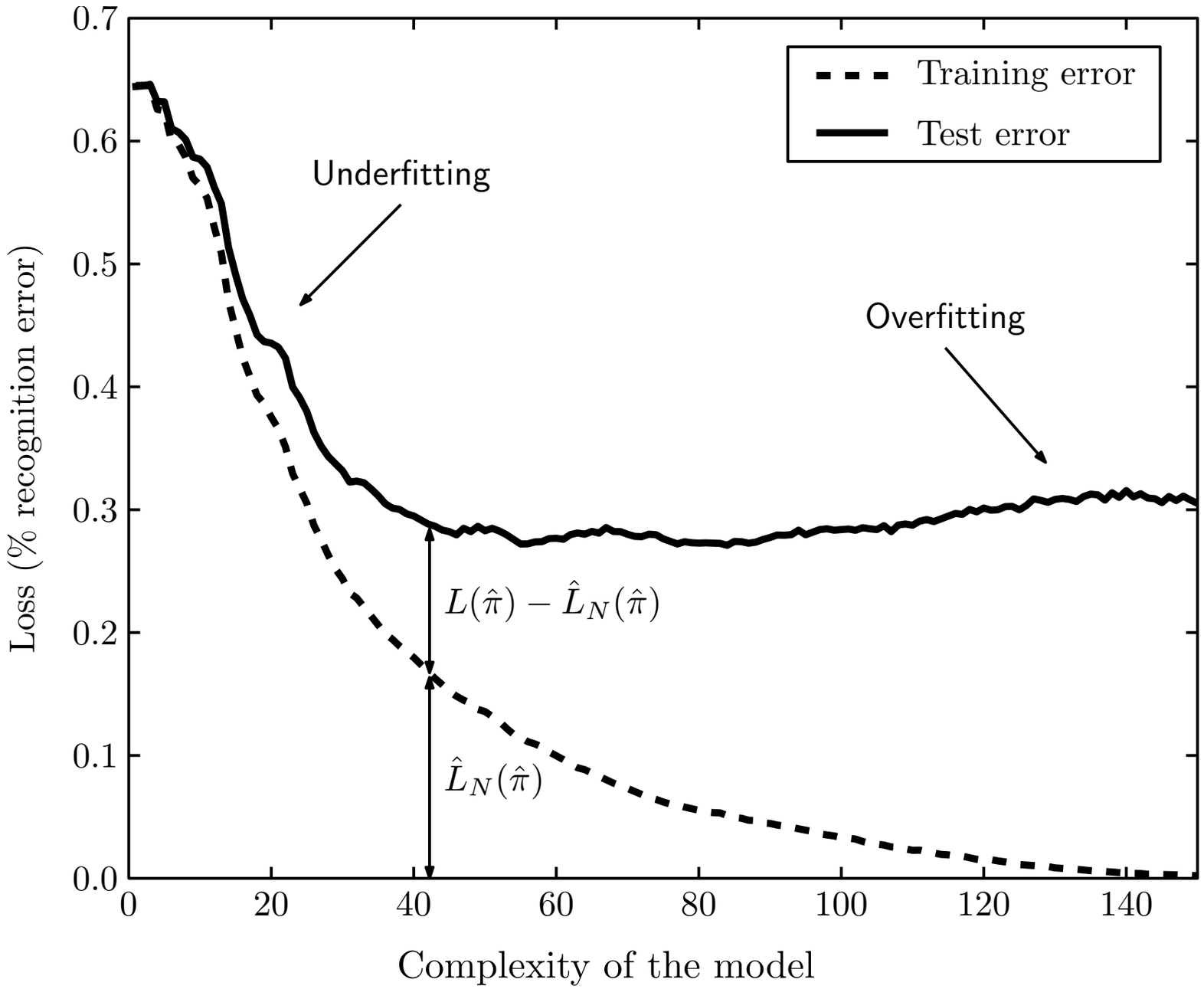
The inductive learning assumption

If we learn from a sufficiently large set of examples, we will do well in the actual task

Overfitting

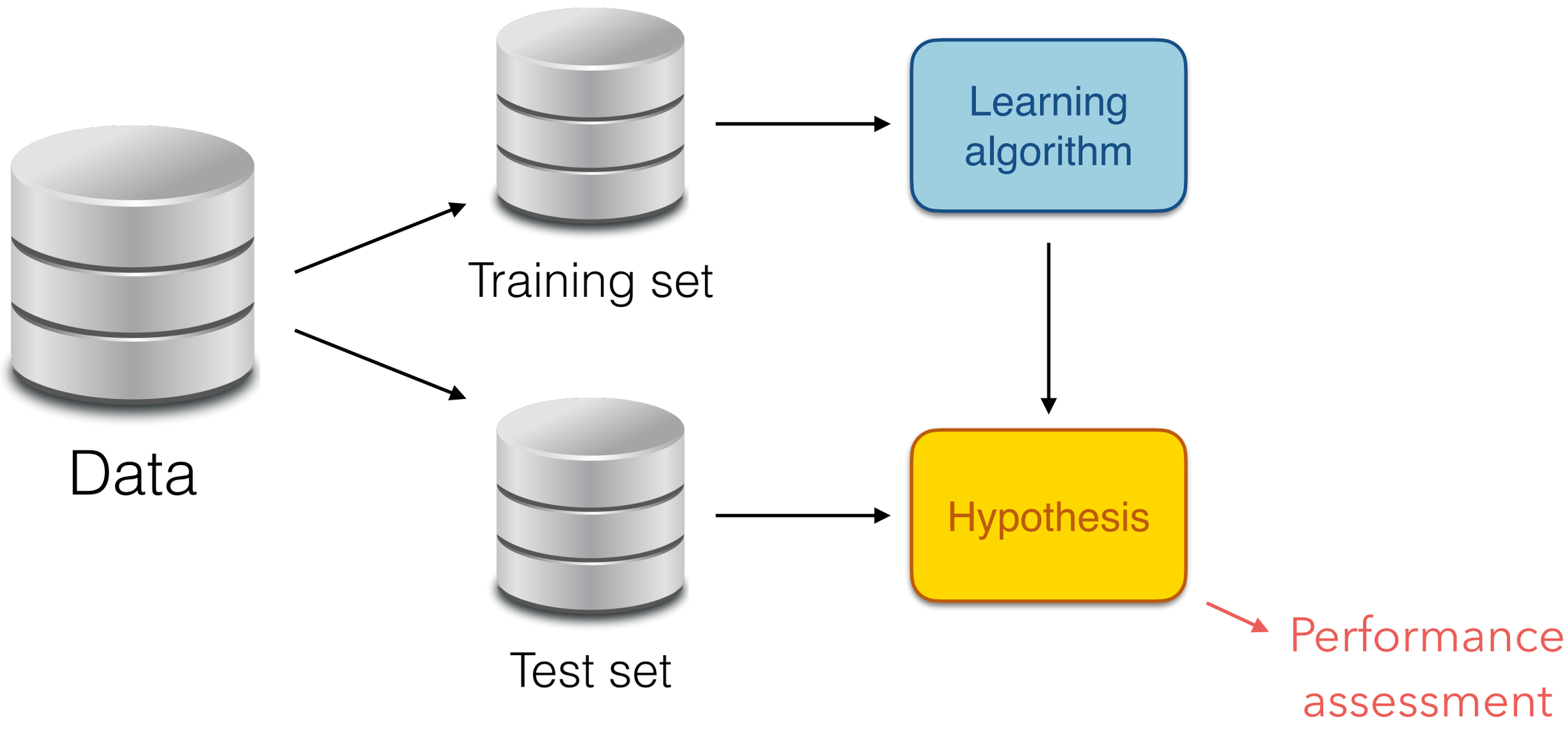
- Set of examples may not be perfect:
  - Overfitting is when our algorithm learns a hypothesis that includes both the **useful information** and **noise** (overfitting)
  - Learning becomes too sensible to small changes in data
  - Two slightly different datasets may learn radically different hypotheses

Overfitting



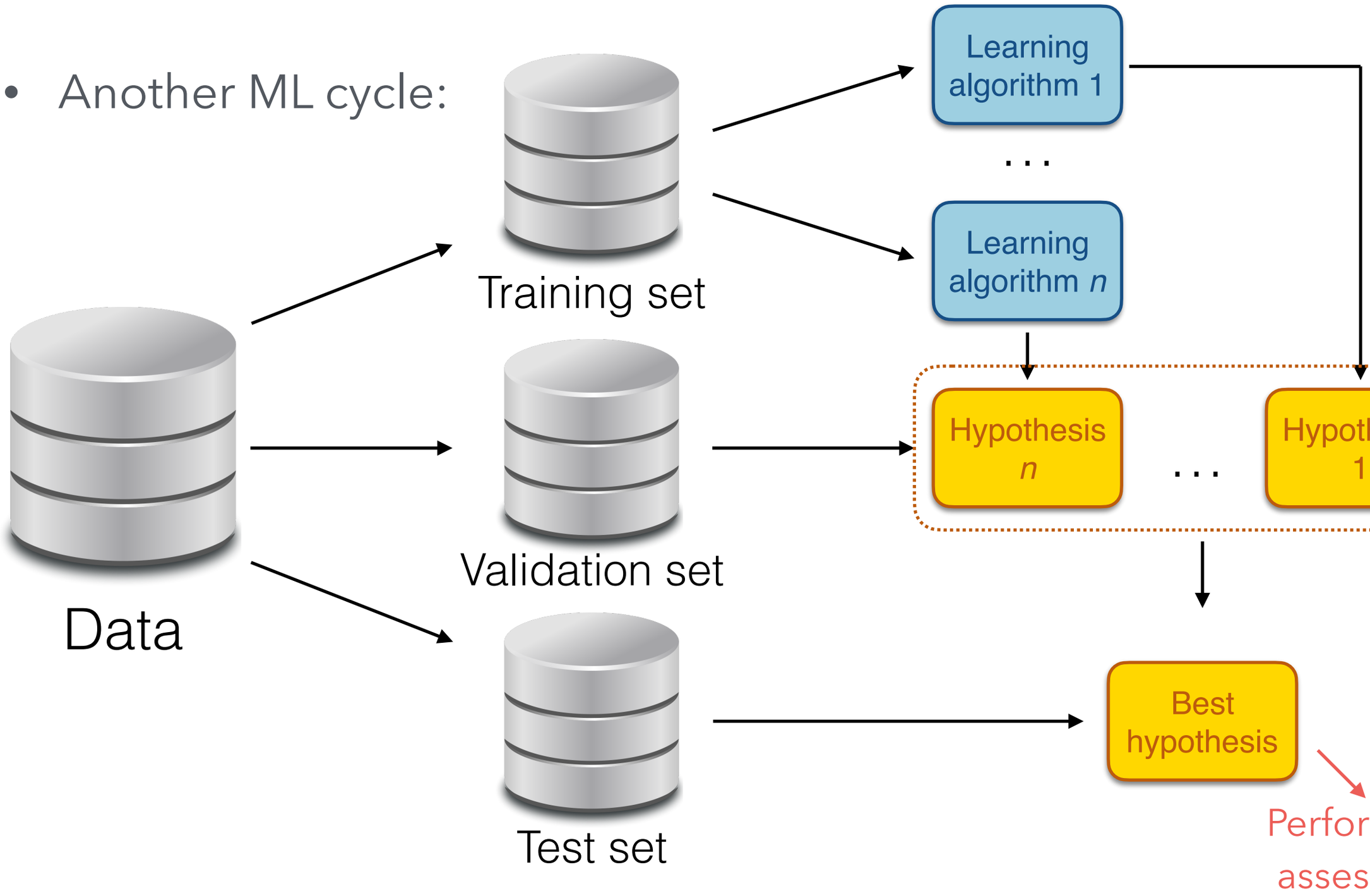
Learning in practice

- The basic learning cycle:



Practice of ML

- Another ML cycle:



Learning rules

Example

- You want to develop a system to automatically identify criminals
- The system is to be deployed in a shopping mall

Example

- The system is to be trained from the following data:

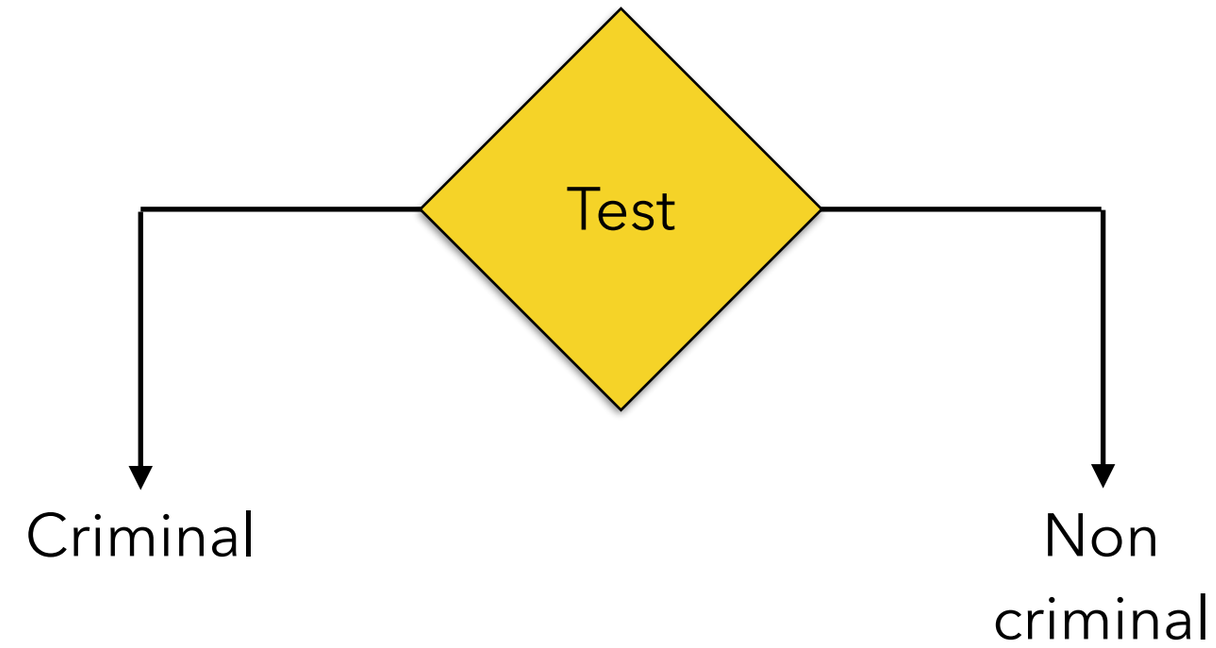
Gun possession	Authority contacts	Movement	Appearance	Criminal
?	Yes	Slowly	Suited	No
No	Yes	Average	Suited	No
?	No	Average	Suited	Yes
Yes	No	Fast	Casual	Yes
?	No	Fast	Rough	Yes
No	No	Slowly	Casual	No
No	No	Fast	Casual	No
?	Yes	Average	Rough	No

Example

- Each individual is described by a set of attributes (*features*)
- These features may be costly to gather
- We want the system to make the best decision using the minimum amount of “tests”

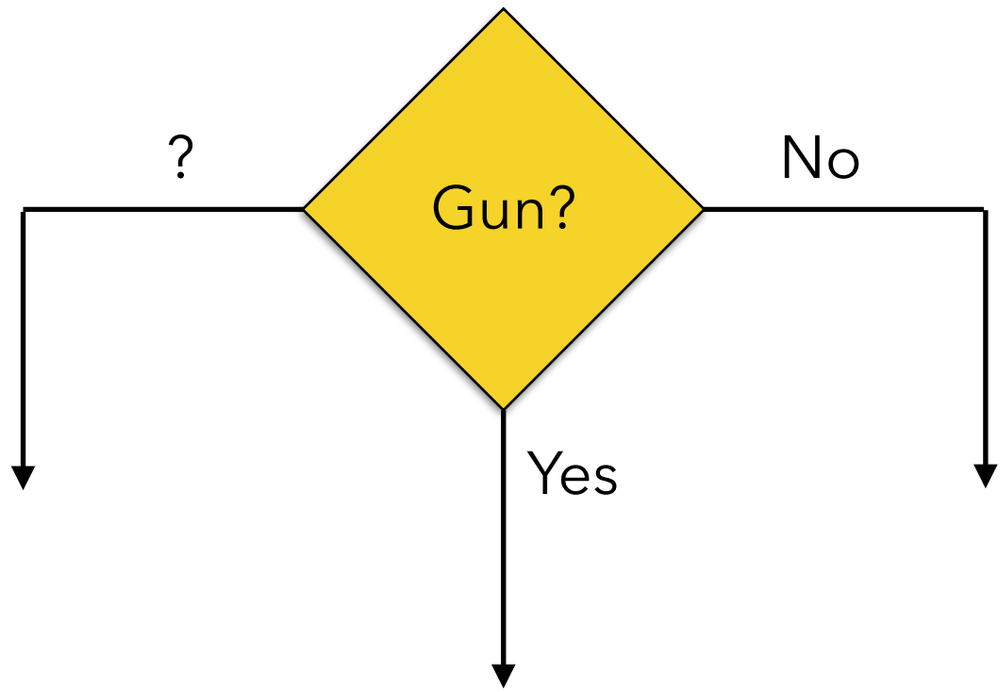
Example

- Ideally:



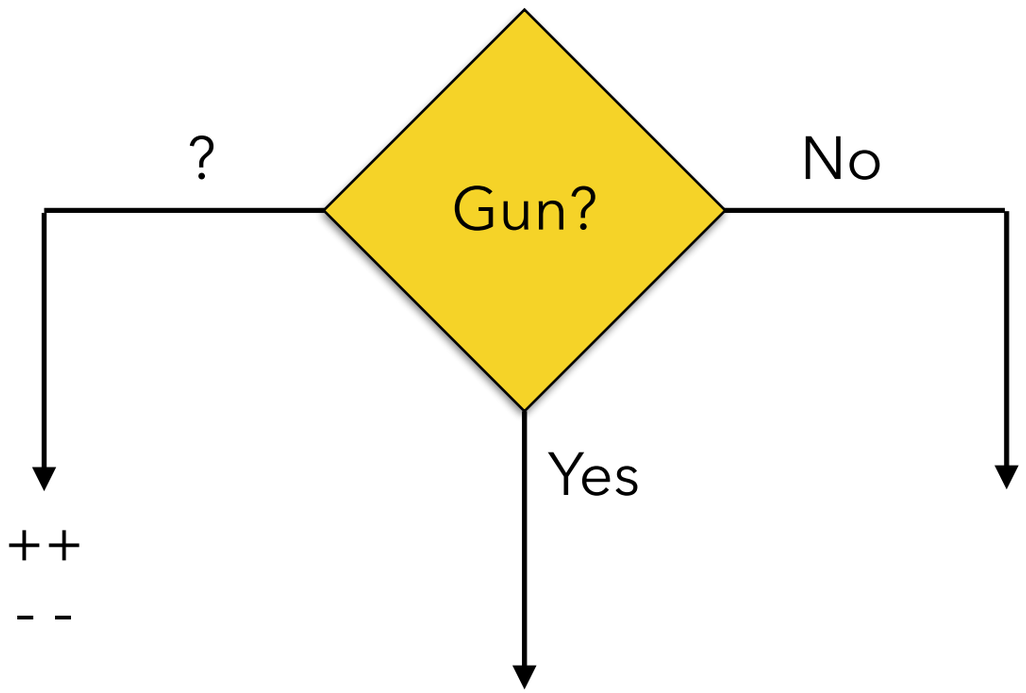
Let's try…

Gun possession	Criminal
?	No
No	No
?	Yes
Yes	Yes
?	Yes
No	No
No	No
?	No



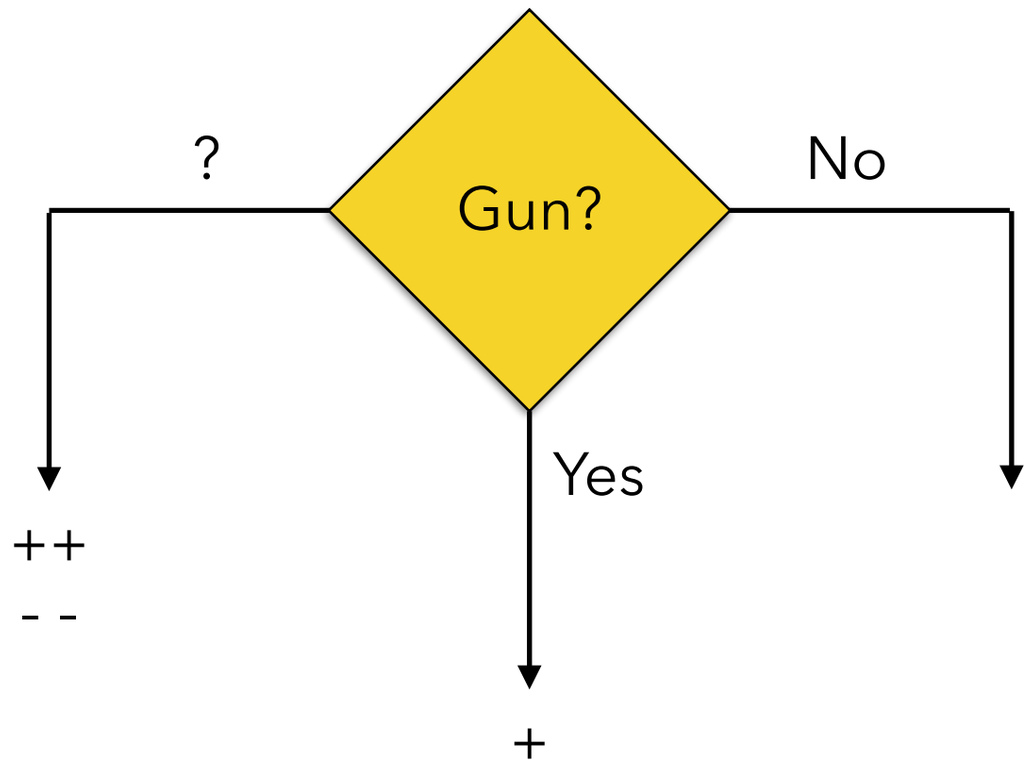
Let's try…

Gun possession	Criminal
?	No
No	No
?	Yes
Yes	Yes
?	Yes
No	No
No	No
?	No



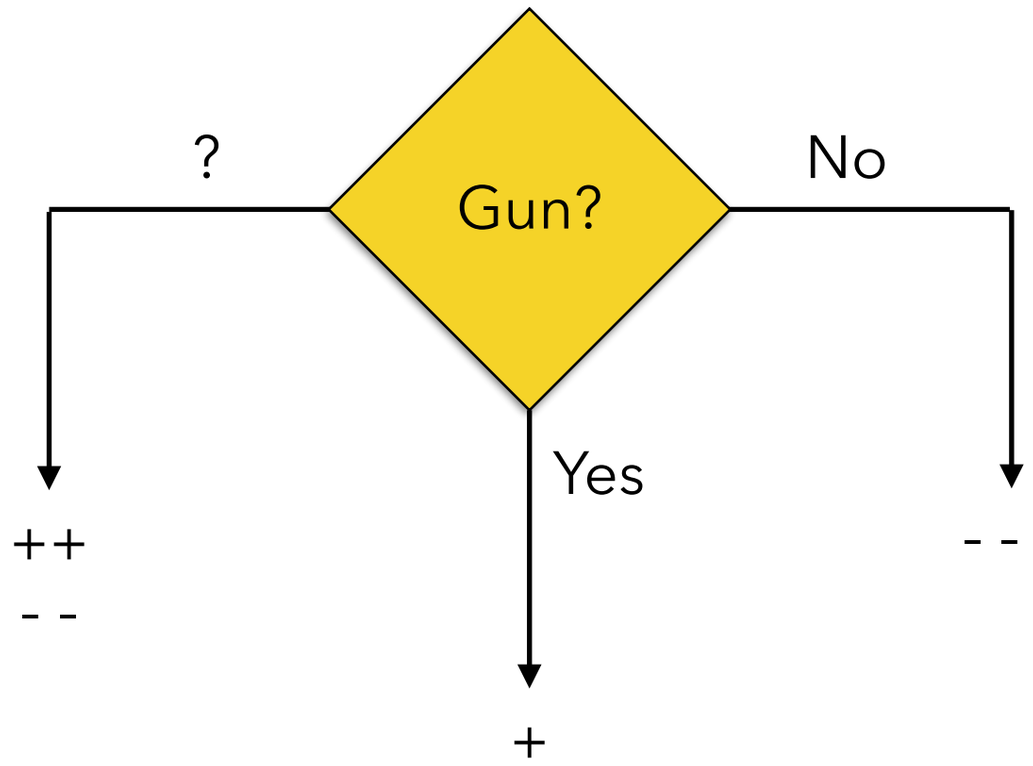
Let's try…

Gun possession	Criminal
?	No
No	No
?	Yes
Yes	Yes
?	Yes
No	No
No	No
?	No



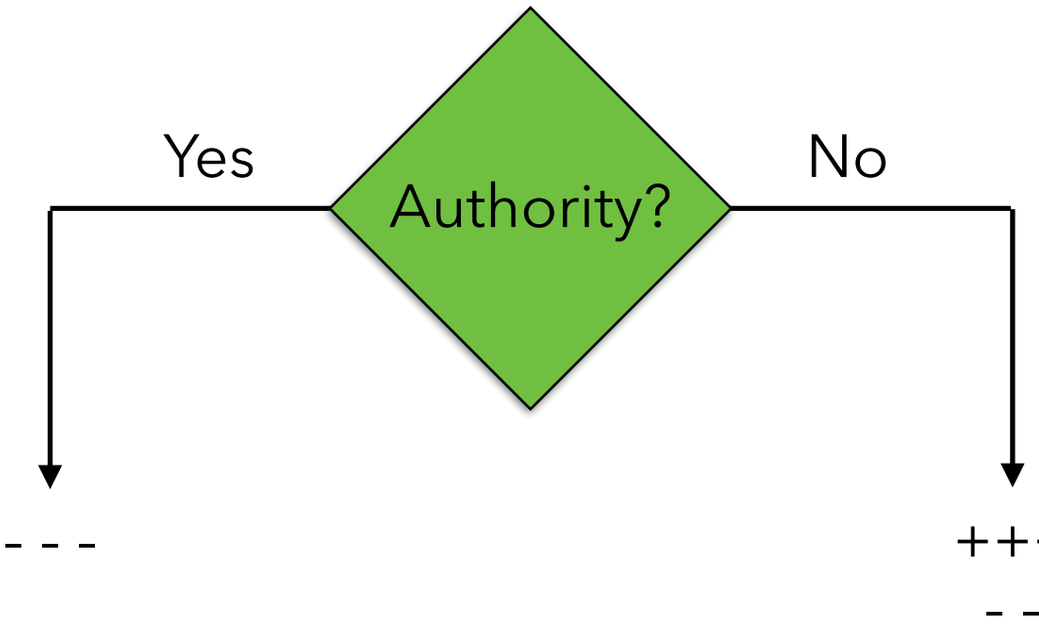
Let's try…

Gun possession	Criminal
?	No
No	No
?	Yes
Yes	Yes
?	Yes
No	No
No	No
?	No



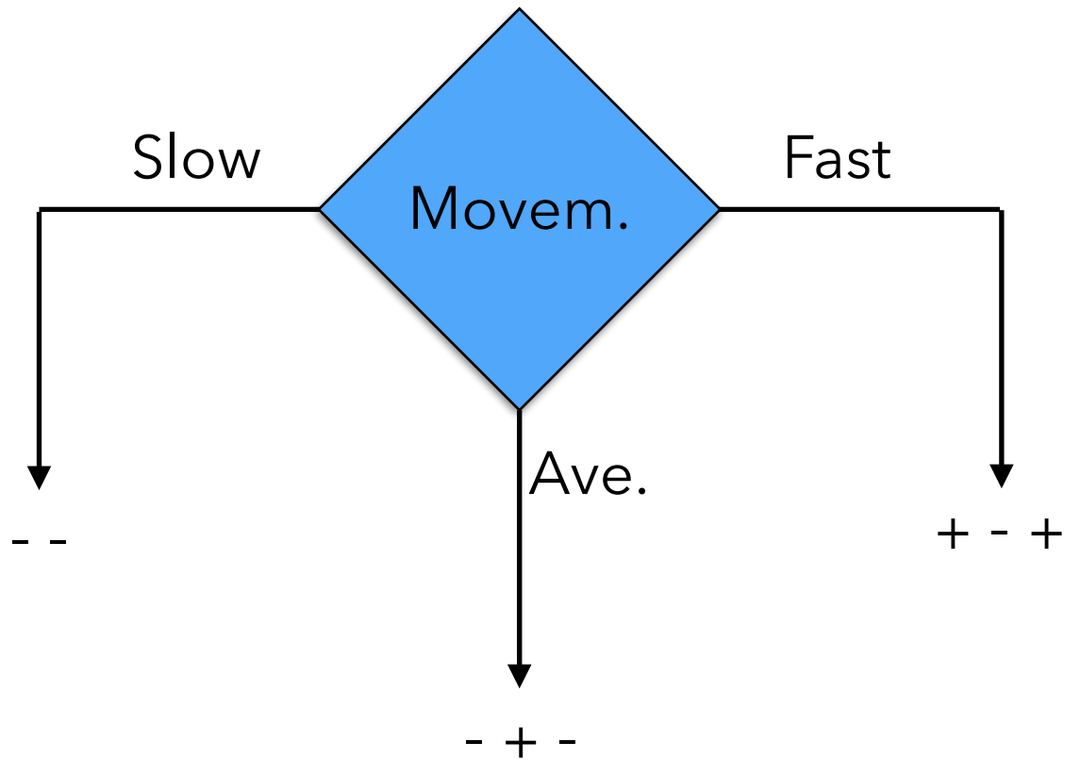
Let's try…

Authority contacts	Criminal
Yes	No
Yes	No
No	Yes
No	Yes
No	Yes
No	No
No	No
Yes	No



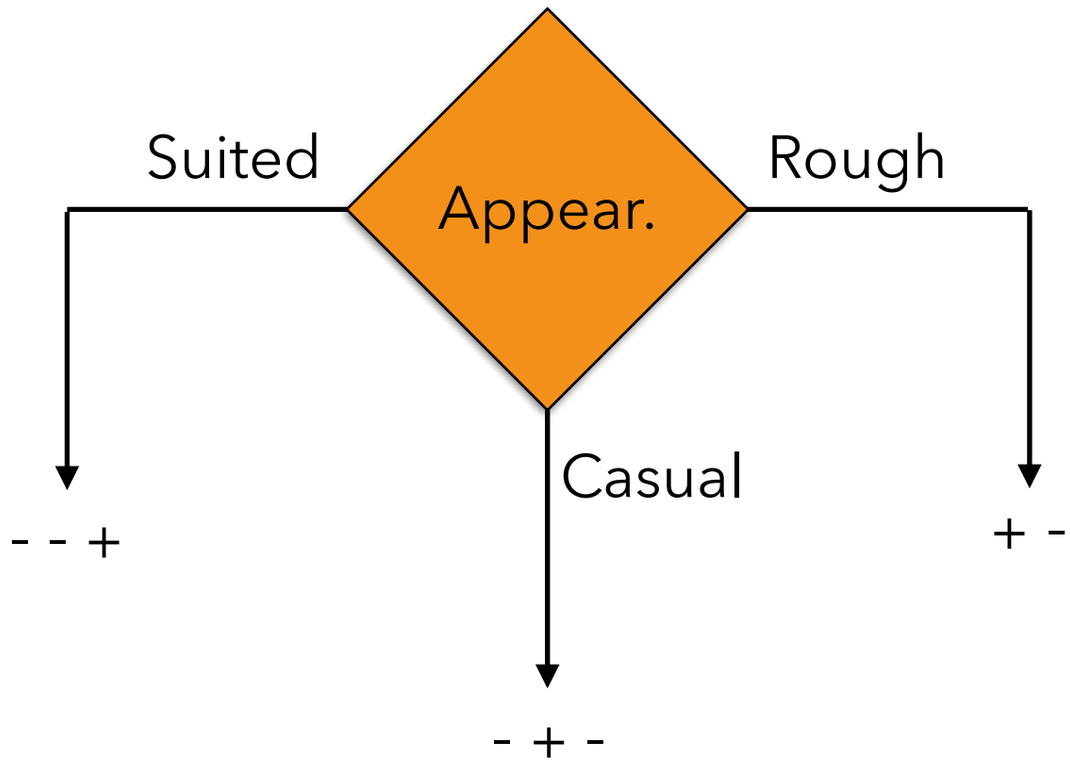
Let's try…

Movement	Criminal
Slowly	No
Average	No
Average	Yes
Fast	Yes
Fast	Yes
Slowly	No
Fast	No
Average	No



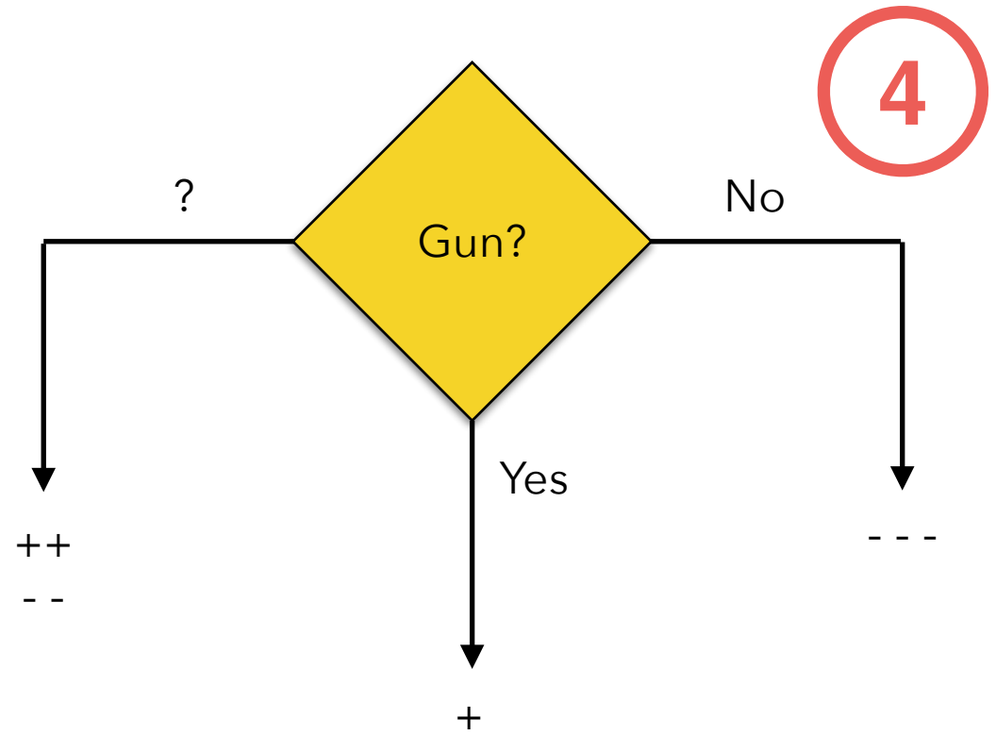
Let's try…

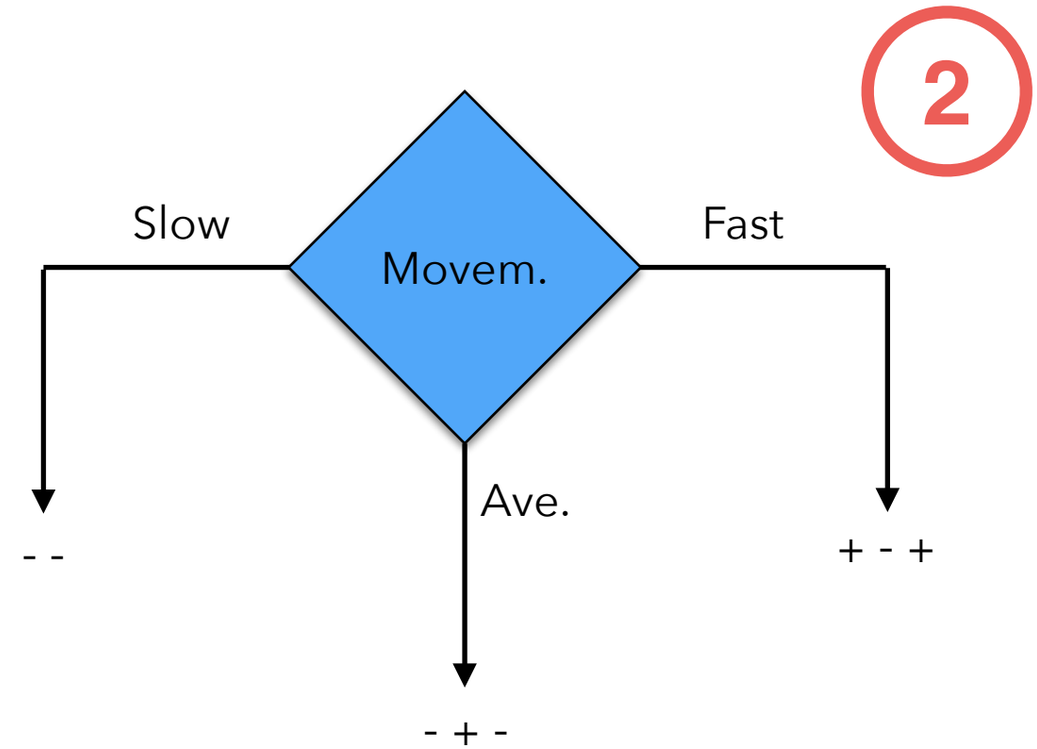
Appearance	Criminal
Suited	No
Suited	No
Suited	Yes
Casual	Yes
Rough	Yes
Casual	No
Casual	No
Rough	No

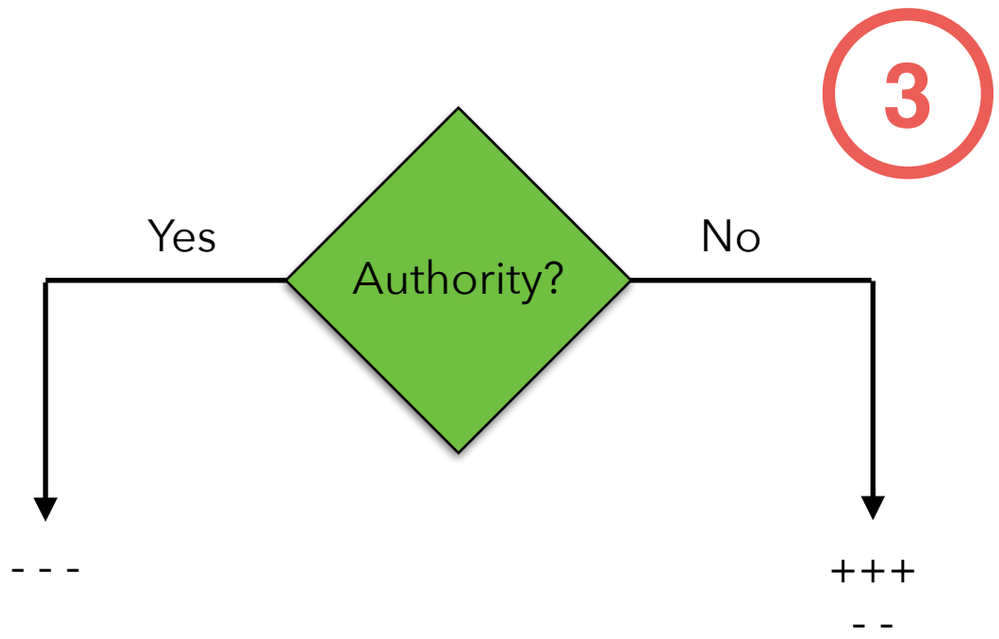


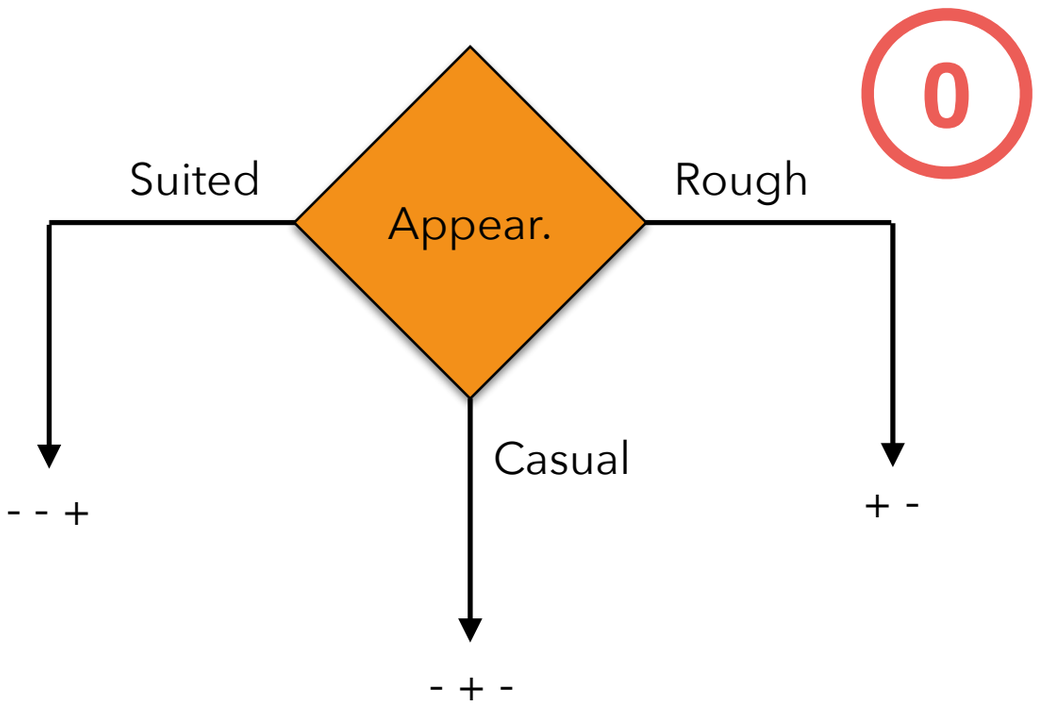
Which is best?

- Let's score each criterion according to how many individuals are placed in "pure" (homogeneous) groups

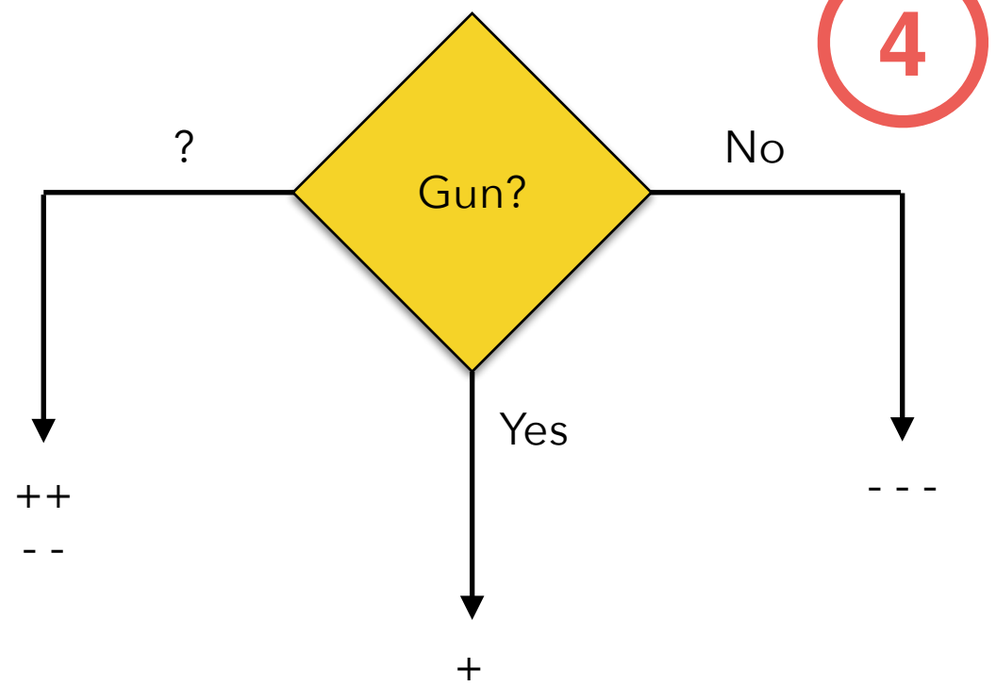

4
Gun?
?
No
Yes
++
- -
- - -
+

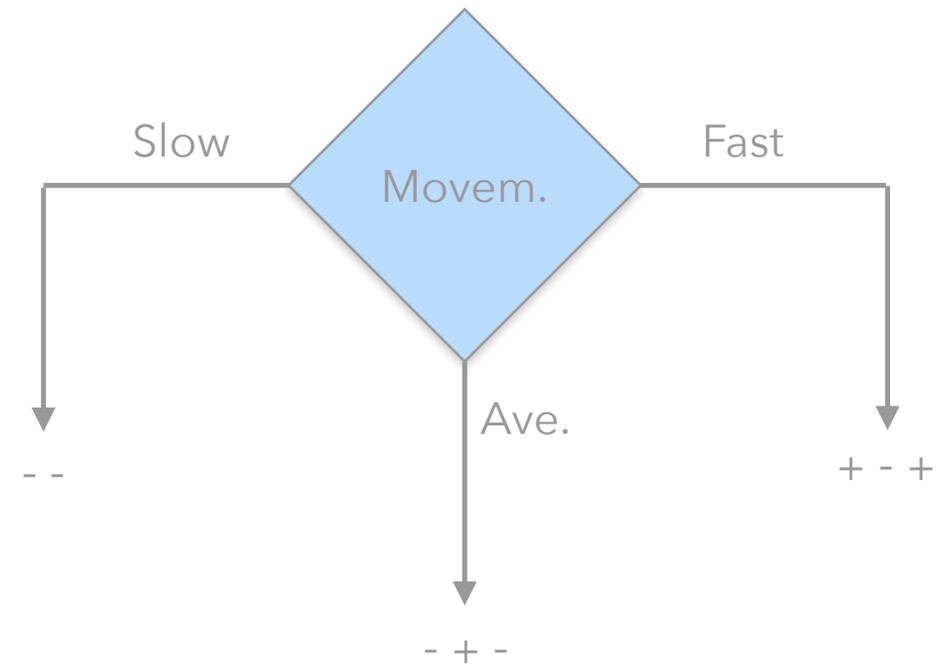

2
Movem.
Slow
Fast
Ave.
- -
+ - +
- + -

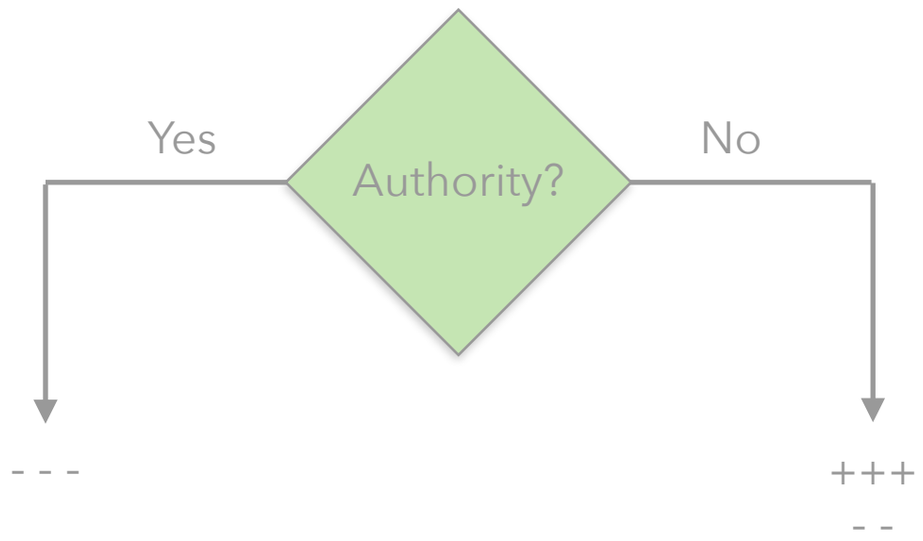

3
Authority?
Yes
No
- - -
+++
- -

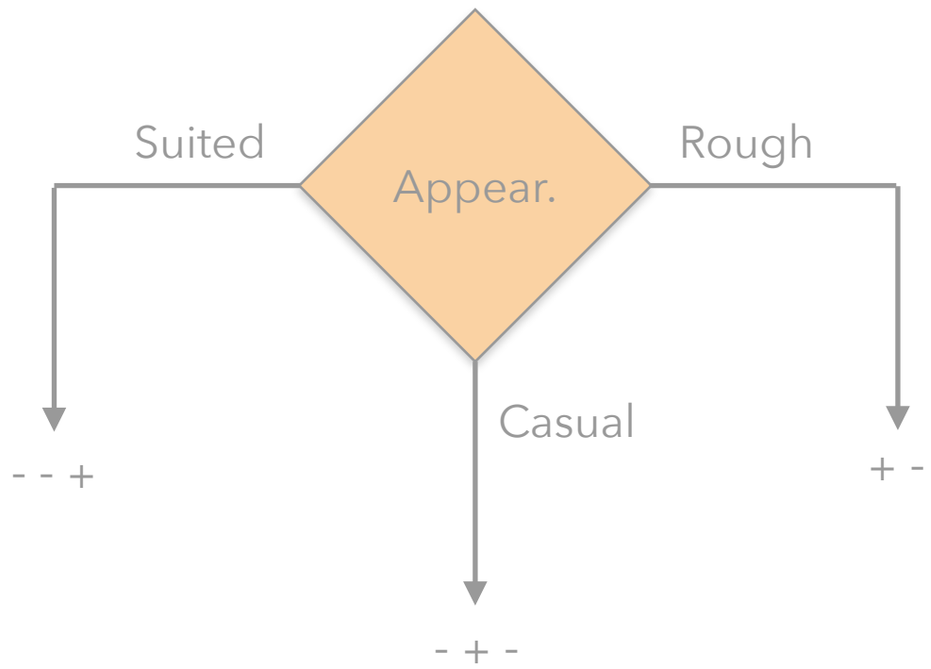

0
Appear.
Suited
Rough
Casual
- - +
+ -
- + -

4


Gun?
?
No
Yes
++
- -
- - -
+

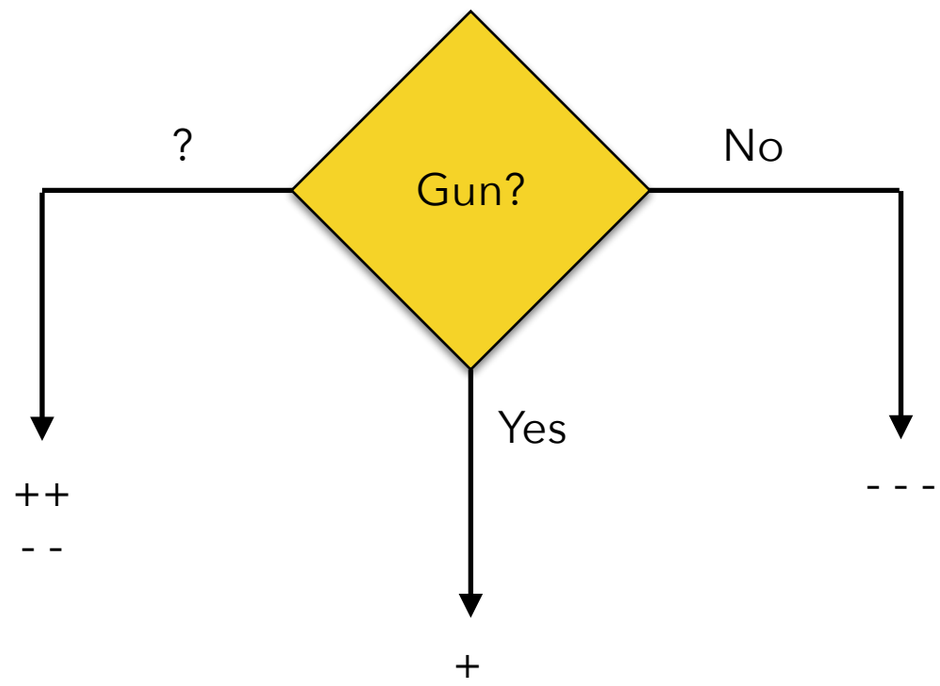

Movem.
Slow
Fast
Ave.
- -
+ - +
- + -


Authority?
Yes
No
- - -
+++
- -


Appear.
Suited
Rough
Casual
- - +
+ -
- + -

This is not enough…

- There is still one group of individuals not properly identified



Individuals not discriminated

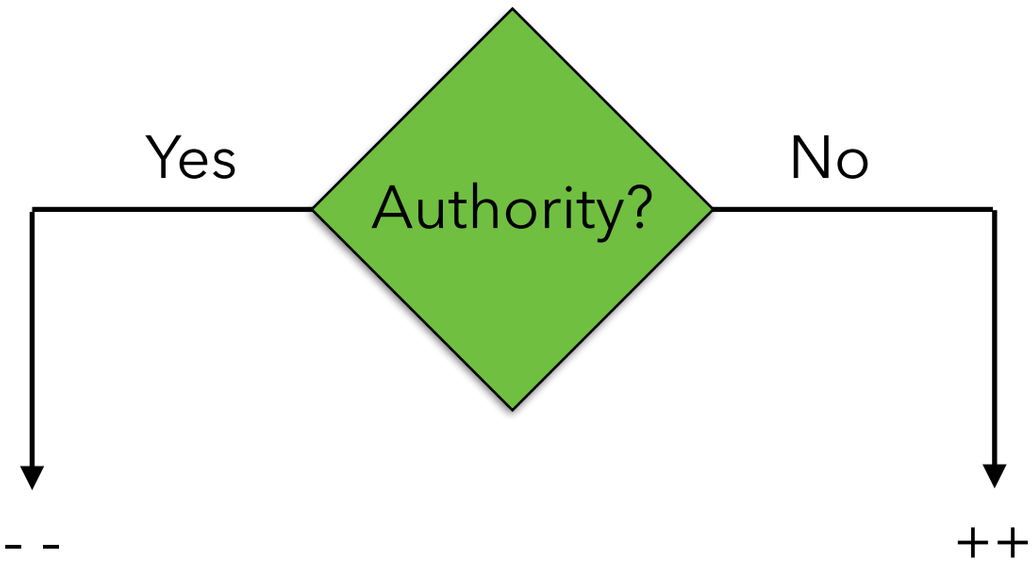
This is not enough…

- There is still one group of individuals not properly identified
- We repeat the process for those individuals

Gun possession	Authority contacts	Movement	Appearance	Criminal
?	Yes	Slowly	Suited	No
No	Yes	Average	Suited	No
?	No	Average	Suited	Yes
Yes	No	Fast	Casual	Yes
?	No	Fast	Rough	Yes
No	No	Slowly	Casual	No
No	No	Fast	Casual	No
?	Yes	Average	Rough	No

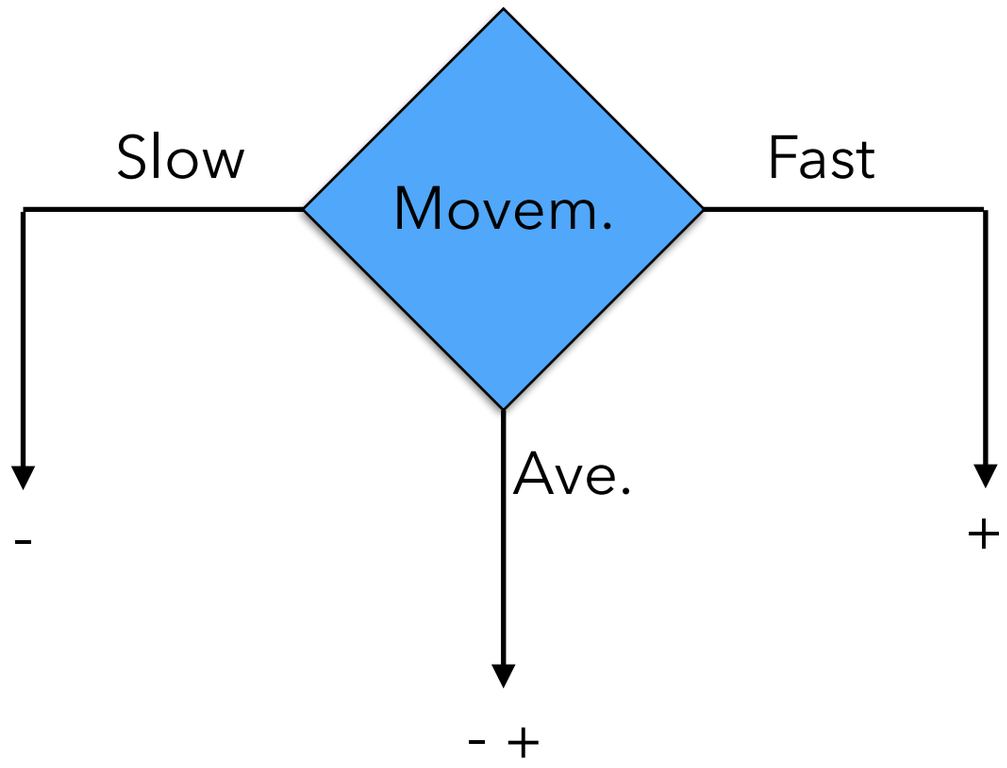
Let's try…

Authority contacts	Criminal
Yes	No
Yes	No
No	Yes
No	Yes
No	Yes
No	No
No	No
Yes	No



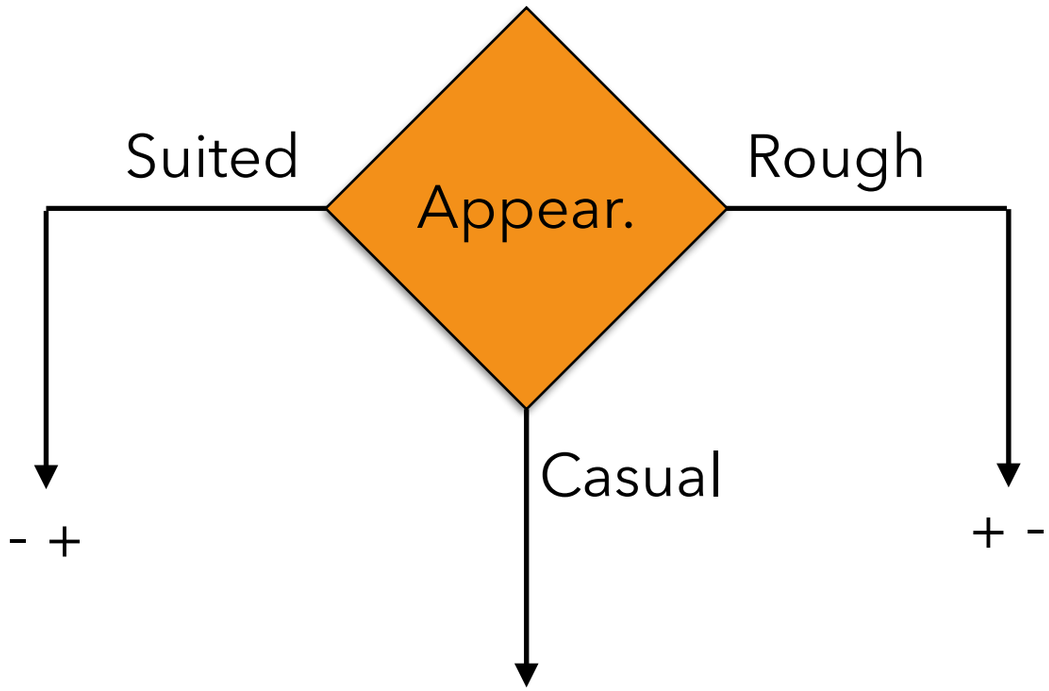
Let's try…

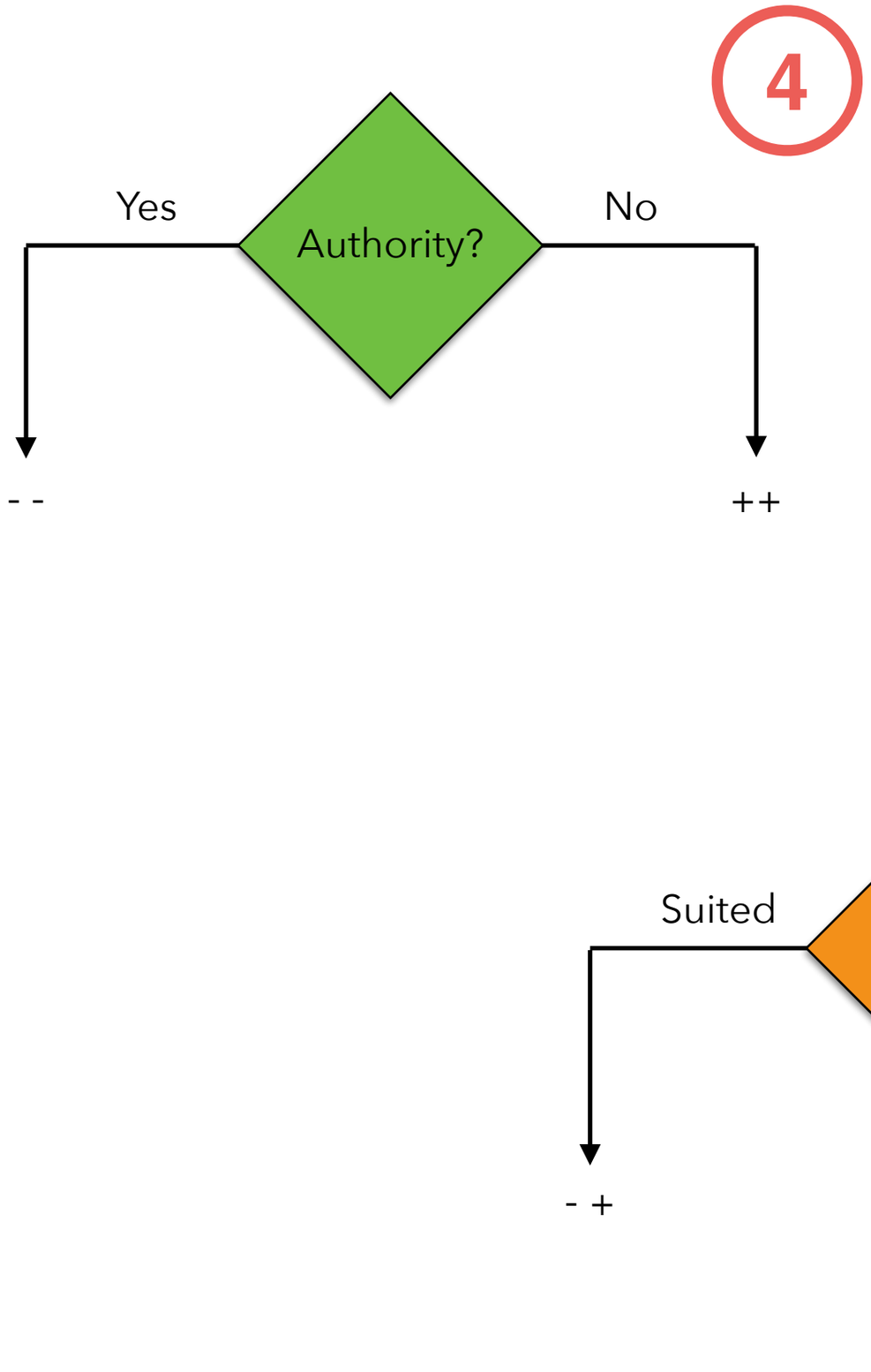
Movement	Criminal
Slowly	No
Average	No
Average	Yes
Fast	Yes
Fast	Yes
Slowly	No
Fast	No
Average	No

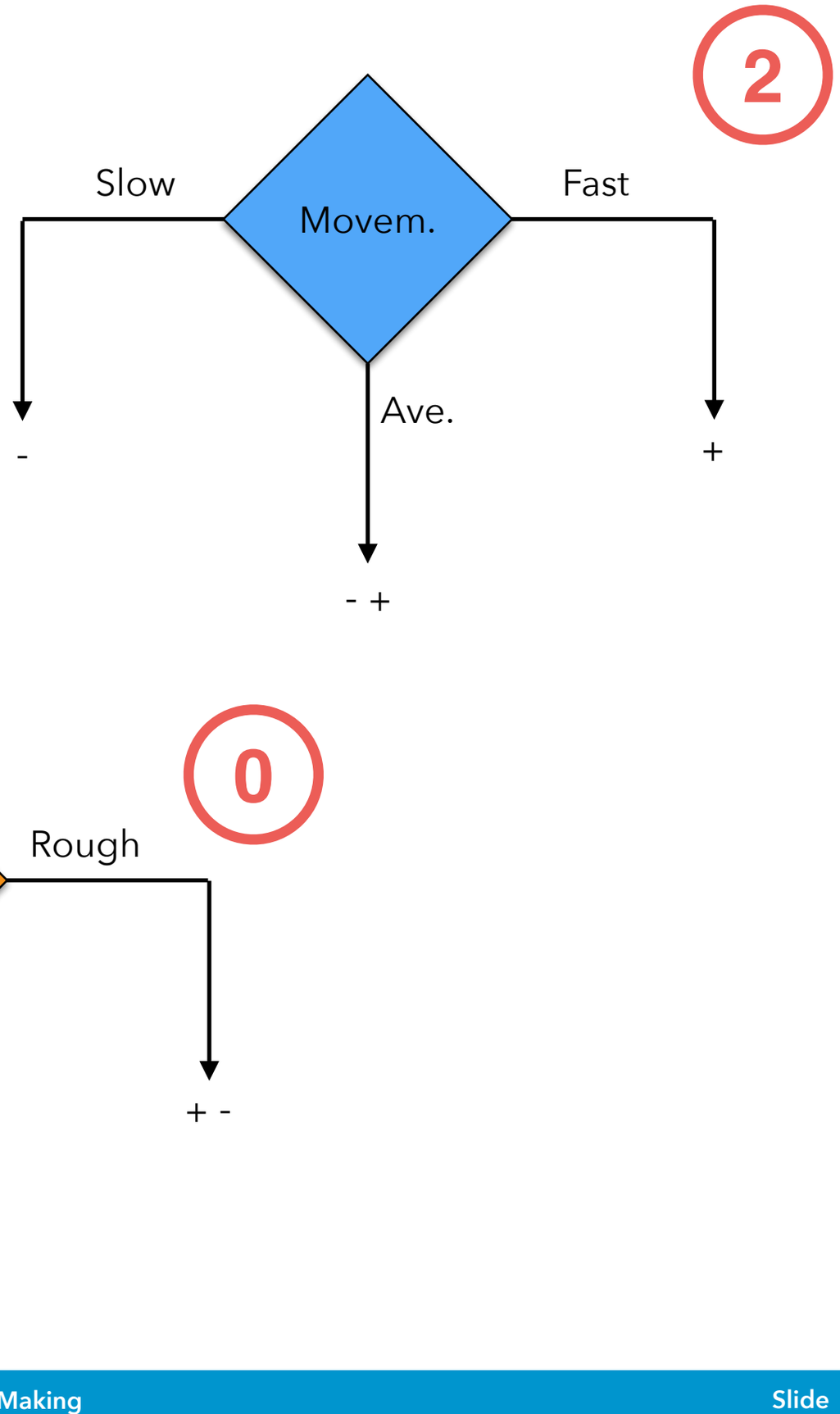


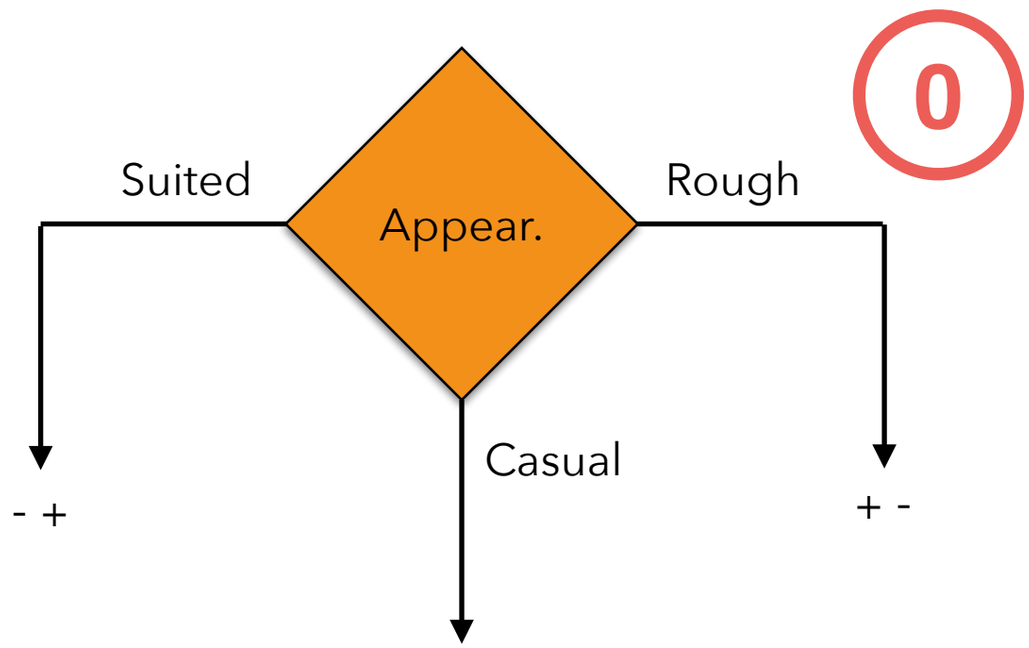
Let's try…

Appearance	Criminal
Suited	No
Suited	No
Suited	Yes
Casual	Yes
Rough	Yes
Casual	No
Casual	No
Rough	No

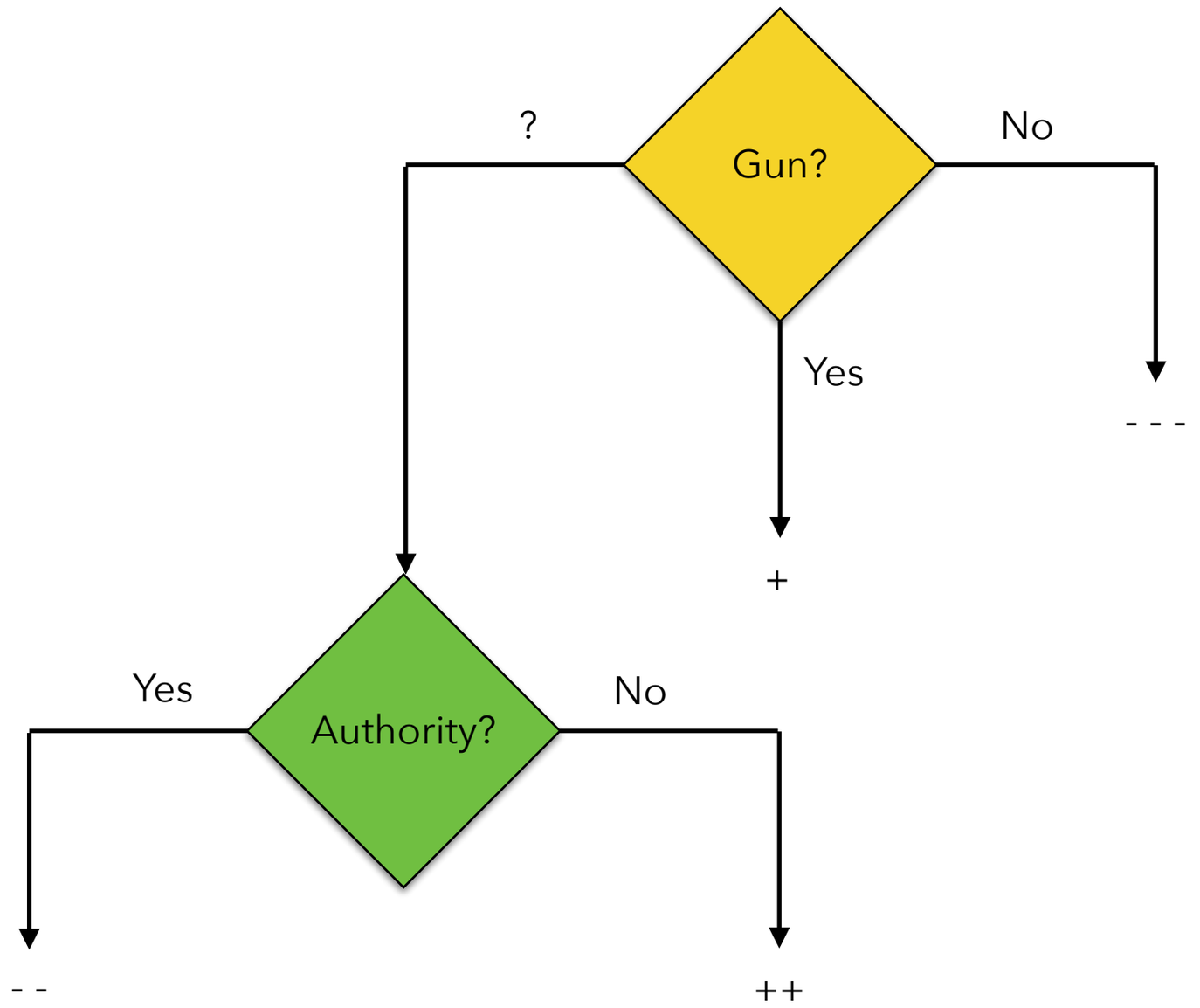



4
Yes
Authority?
No
- -
++


2
Slow
Movem.
Fast
Ave.
-
+
- +


0
Suited
Appear.
Rough
Casual
- +
+ -

Finally...



Problem…

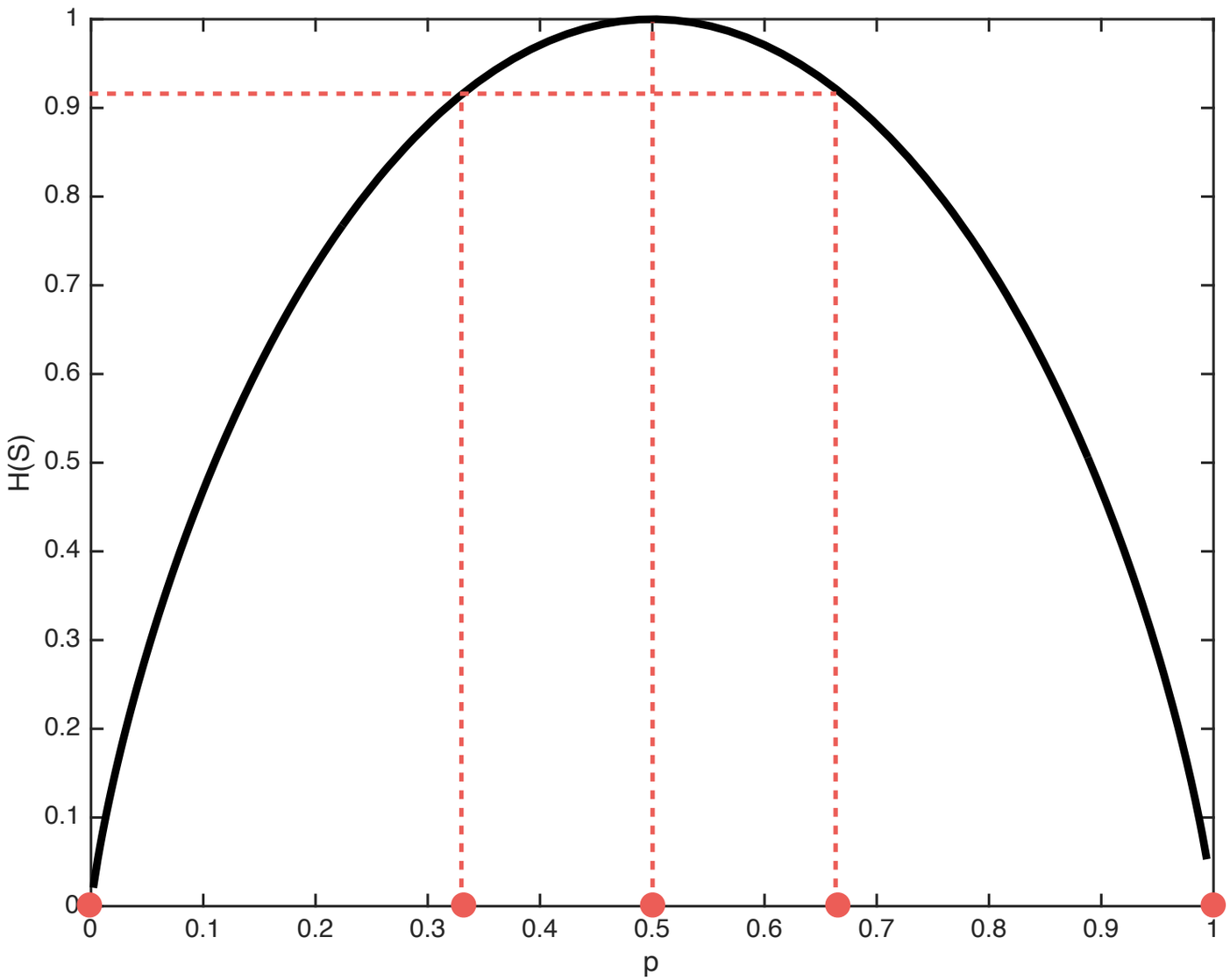
- What if you have 10,000 individuals?
  - There won’t be any criterion with individuals placed in “pure” groups
  - All criteria will score 0!

Measuring "disorder"

- We can instead try to measure the "disorder" in the resulting sets
  - Disorder in information theory: **entropy**
  - Given a set *S* of *N* bits, let
    - *p* – amount of 1s
    - *n* – amount of 0s
- The entropy of set *S* is, then

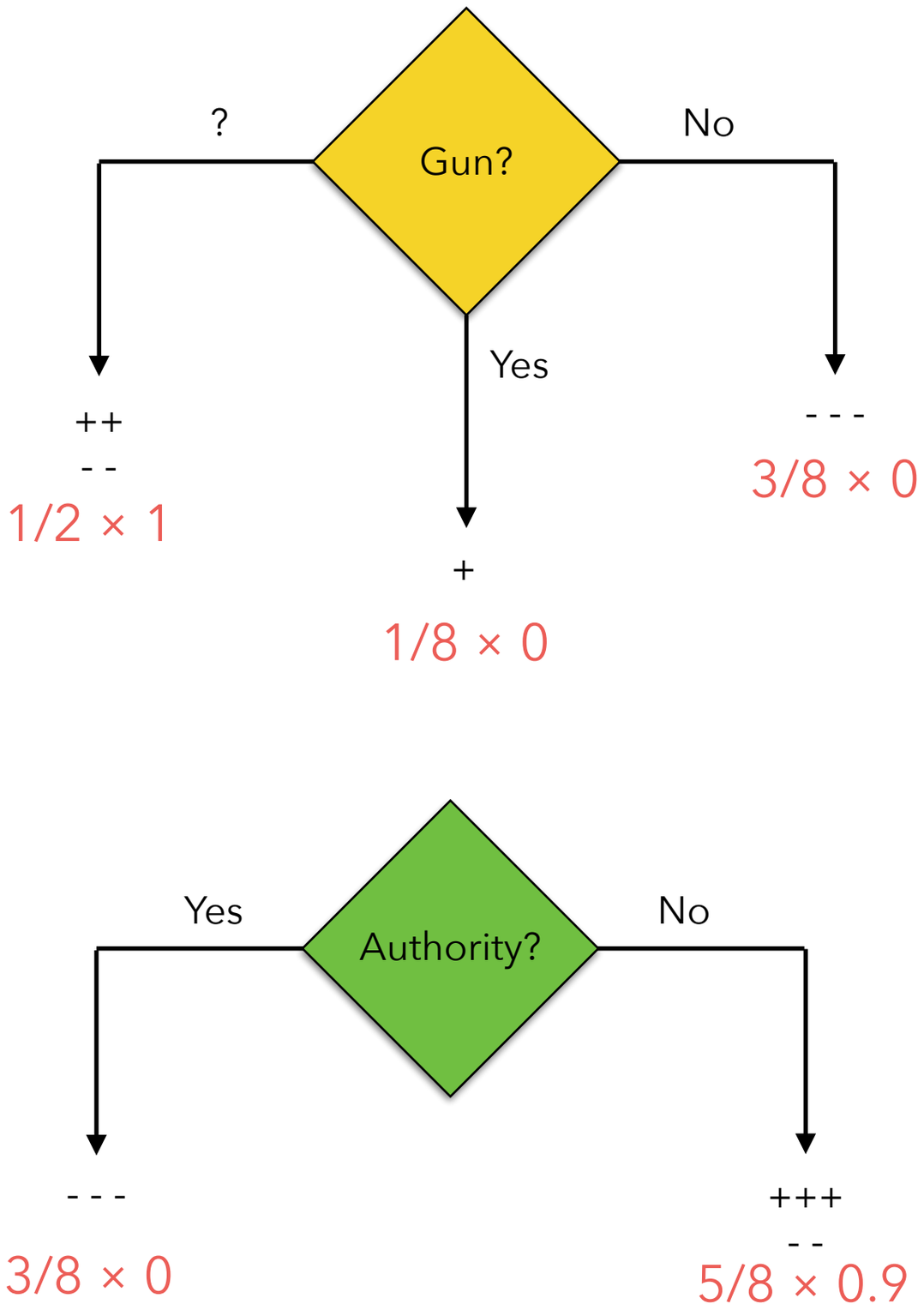
$$H(S) = -\frac{p}{N}\log_2\frac{p}{N} - \frac{n}{N}\log_2\frac{n}{N}$$

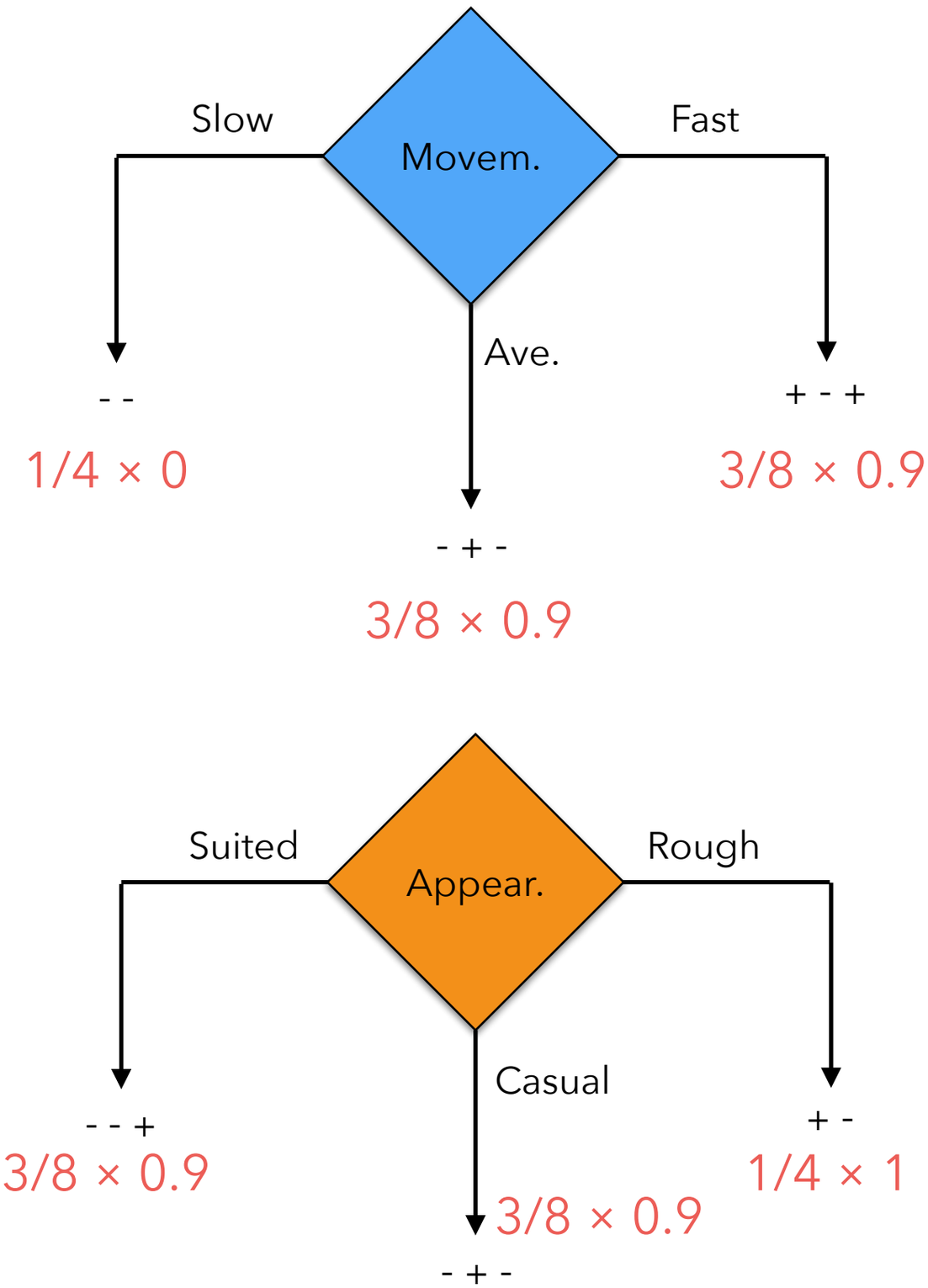
Entropy



Measuring "disorder"

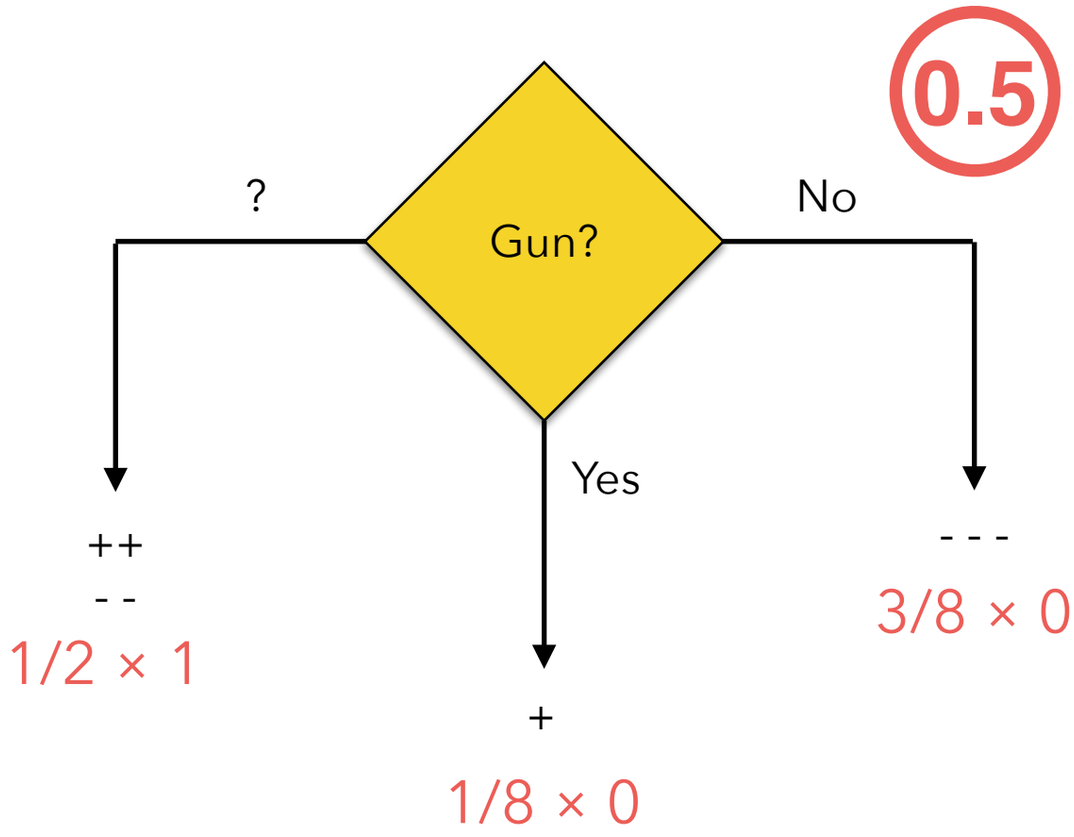
- For each of the criteria, we measure the "average entropy" of the resulting sets

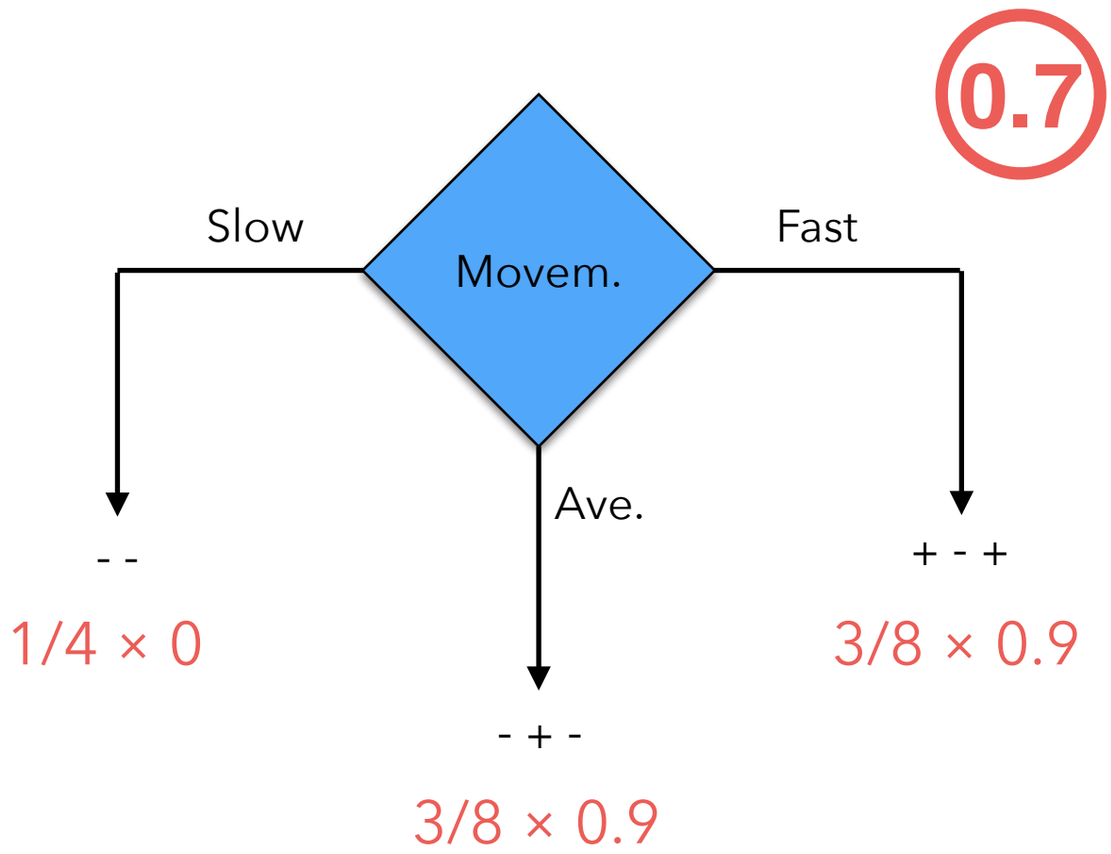

Gun?
?
No
Yes
++
- -
1/2 × 1
+
1/8 × 0
- - -
3/8 × 0

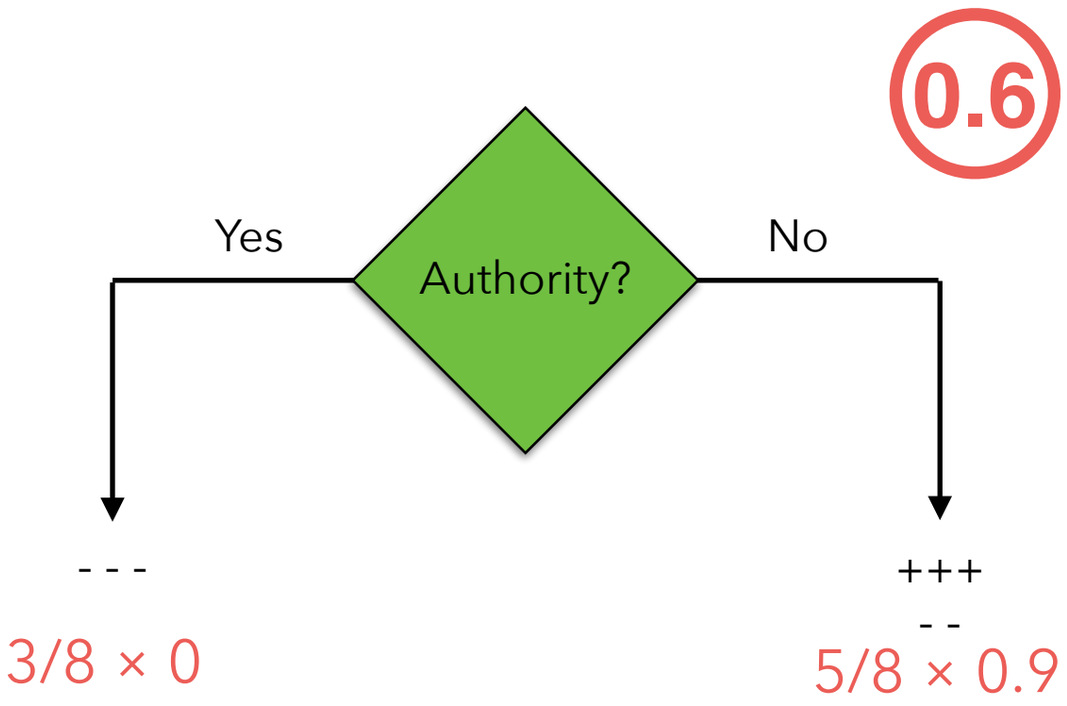

Movem.
Slow
Fast
Ave.
- -
1/4 × 0
- + -
3/8 × 0.9
+ - +
3/8 × 0.9

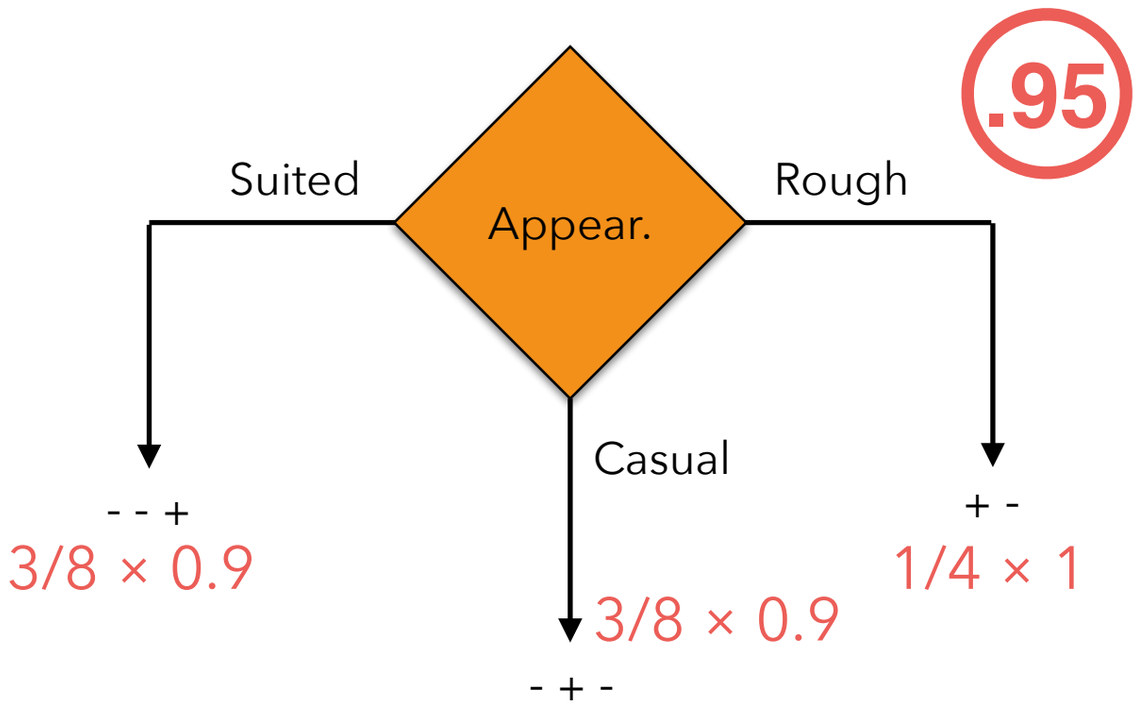
Authority?
Yes
No
- - -
3/8 × 0
+++
- -
5/8 × 0.9

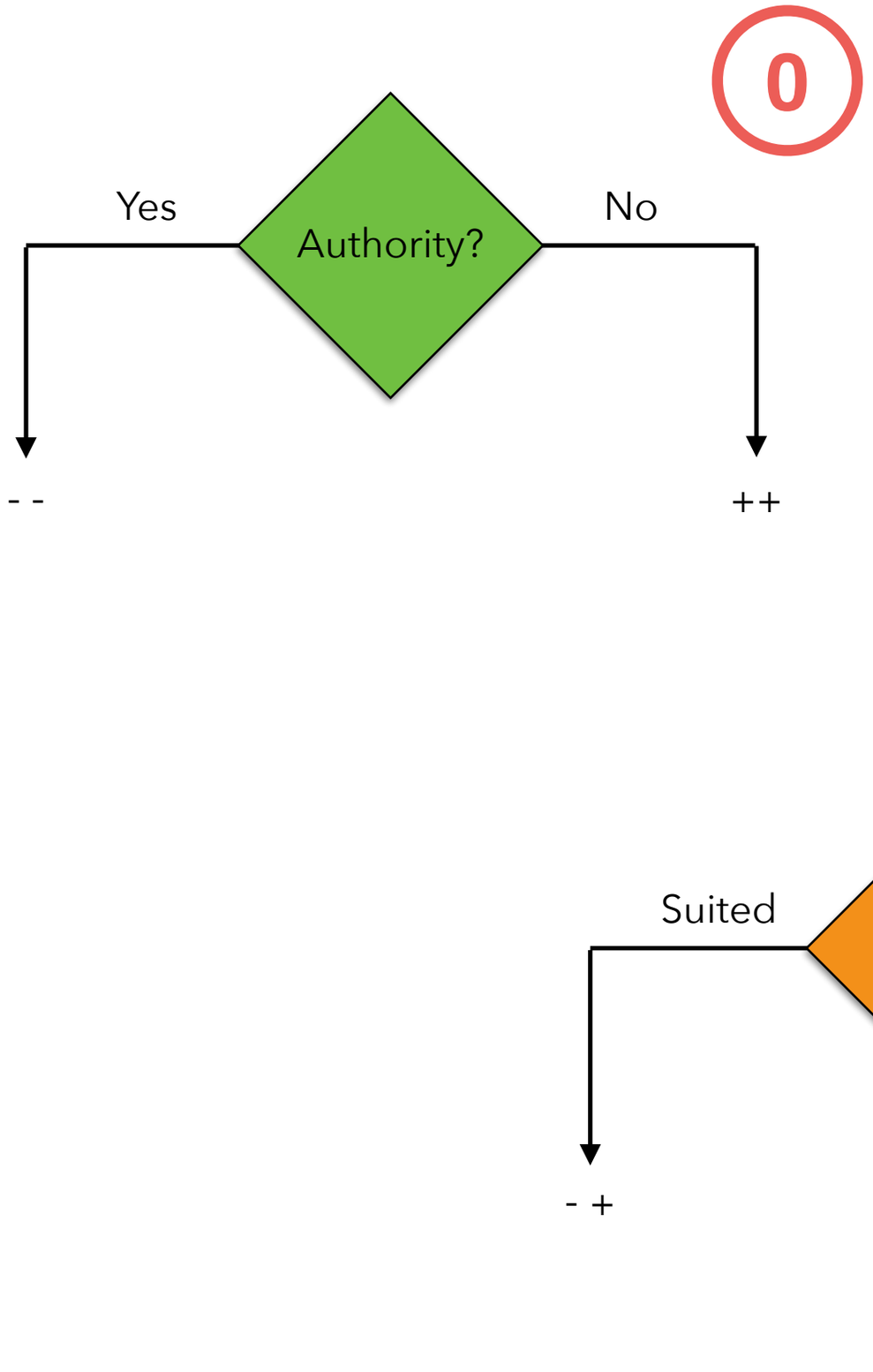
Appear.
Suited
Rough
Casual
- - +
3/8 × 0.9
+ -
1/4 × 1
- + -
3/8 × 0.9

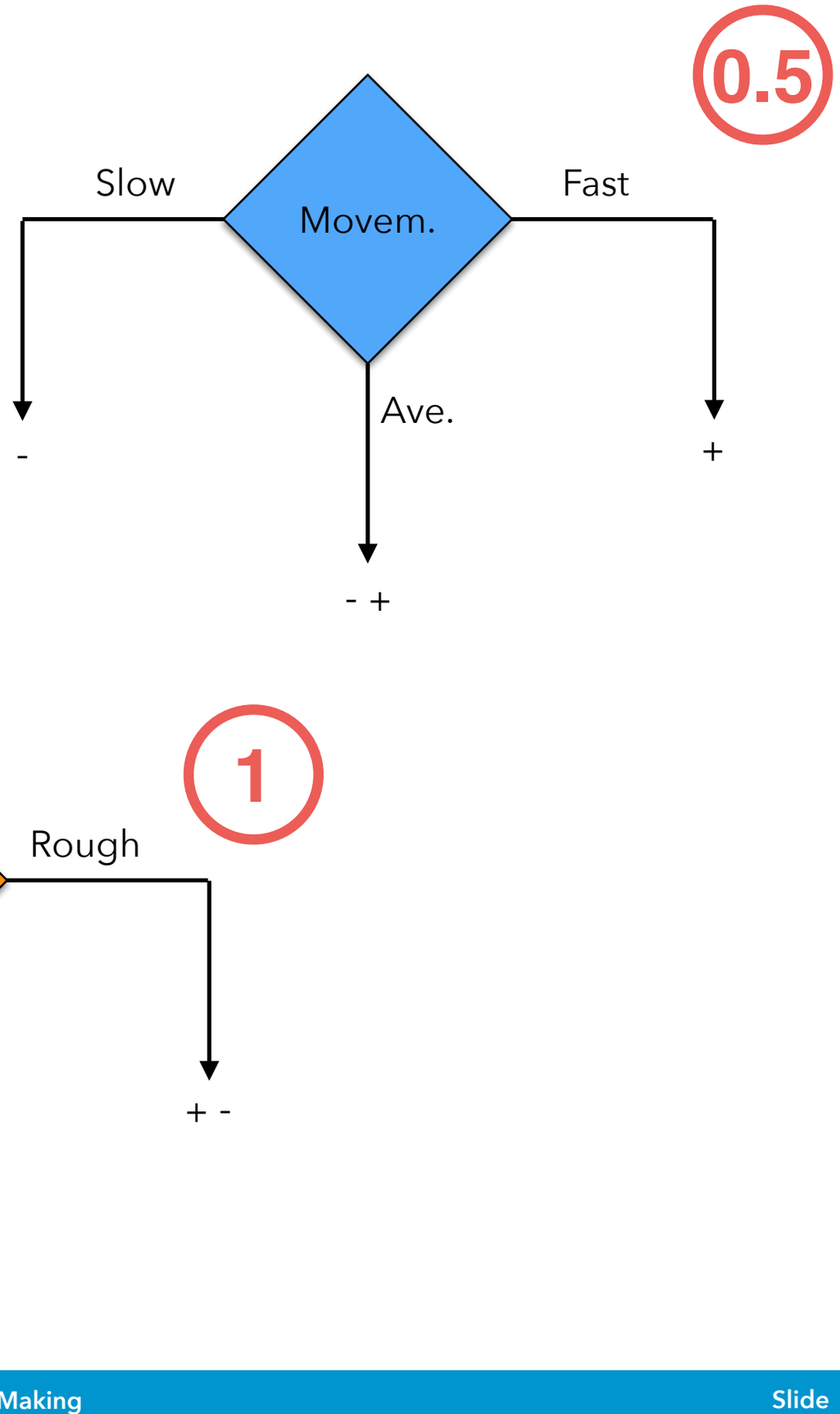

0.5
Gun?
?
No
Yes
++
- -
1/2 × 1
+
1/8 × 0
- - -
3/8 × 0

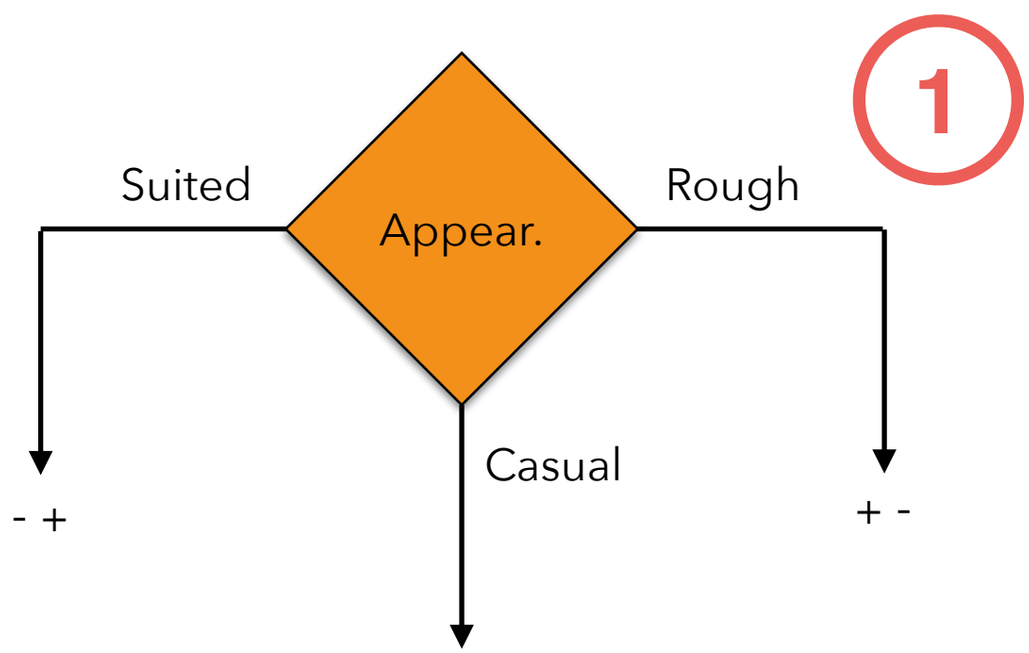

0.7
Movem.
Slow
Fast
Ave.
- -
1/4 × 0
- + -
3/8 × 0.9
+ - +
3/8 × 0.9


0.6
Authority?
Yes
No
- - -
3/8 × 0
+++
- -
5/8 × 0.9

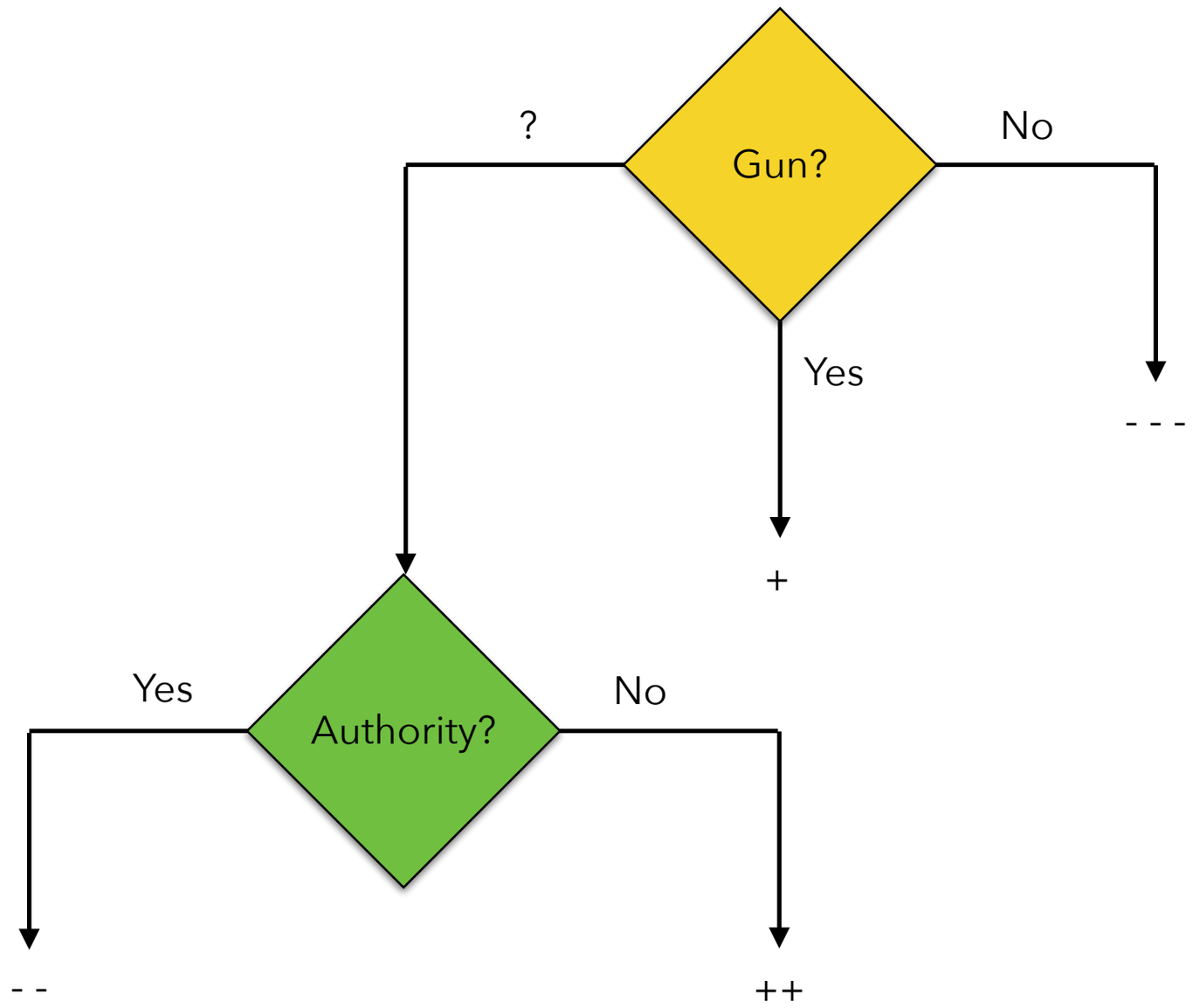

.95
Appear.
Suited
Rough
Casual
- - +
3/8 × 0.9
+ -
1/4 × 1
- + -
3/8 × 0.9


0
Yes
Authority?
No
- -
++


0.5
Slow
Movem.
Fast
Ave.
-
+
- +

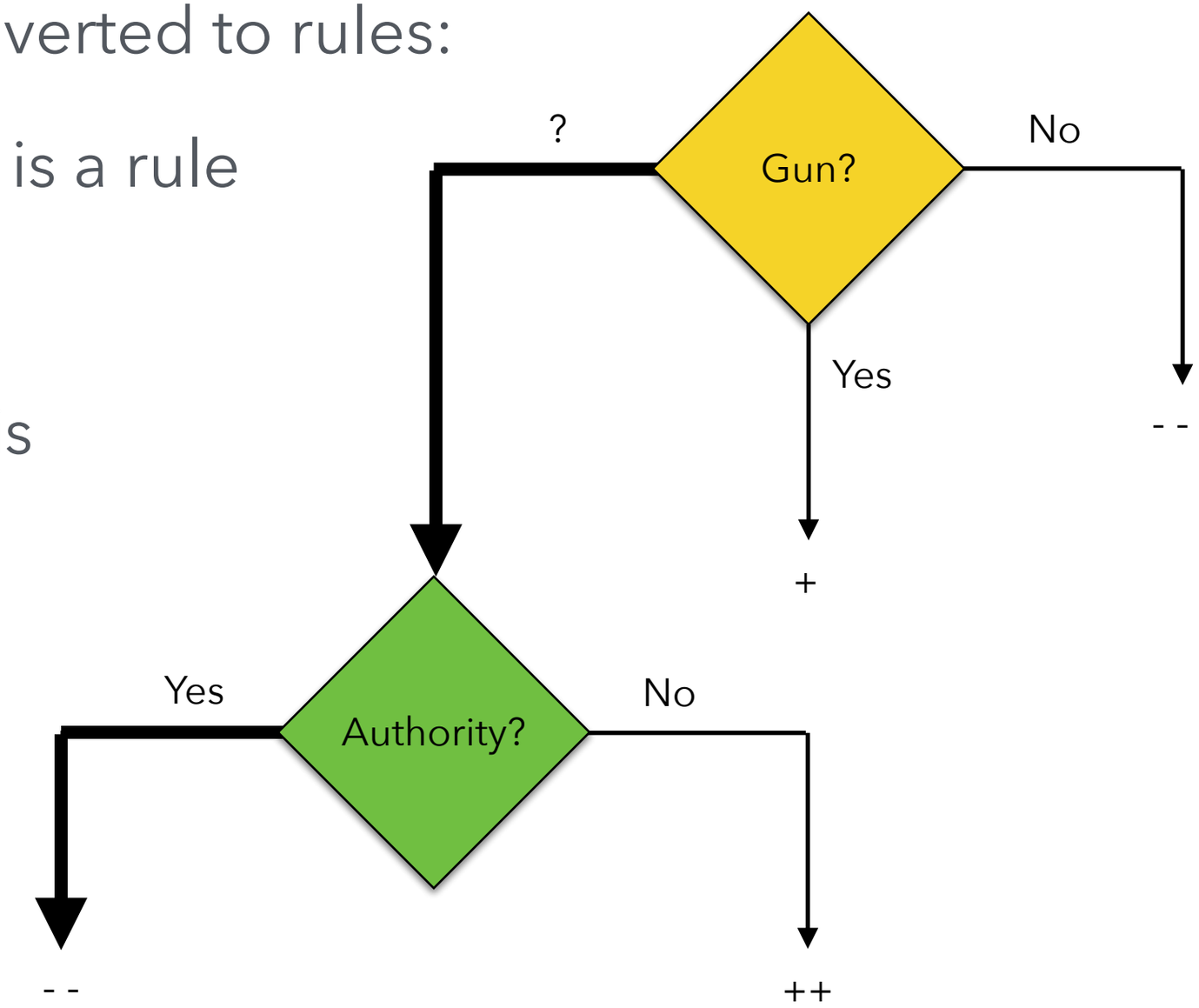

1
Suited
Appear.
Rough
Casual
- +
+ -

Finally...



Converting trees to rules

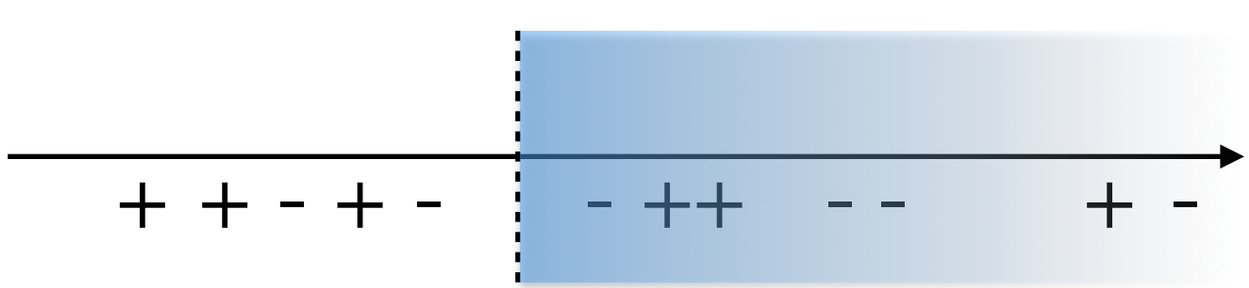
- Trees can be trivially converted to rules:
  - Each path root → leaf is a rule
  - E.g.,
    - "If gun possession is unknown and the person talks to the authority, then it is not a criminal"



To conclude…

Numerical attributes

- What if some of the attributes are numerical?
  - Create a binary test: is attribute > threshold?



- Which threshold?
  - Try all values between data-points

Example