How Can We Tell When an AI Is Really Smart — and When It Is Just Fooling Us?

An Illustrated Plain-Language Guide to Detect BS — The Mathematics

Based on Detect BS - The Mathematics, Version 1.2.1 (16 August 2026)

Every section has one picture drawn for that section alone, and a foldout flowchart of the whole procedure on the next page.

Before You Start

Imagine someone tells you, “I built a computer that can predict what will happen tomorrow!” Your first question might be, “How often is it right?” That is a good question, but it is not enough. A computer can get a good score for a bad reason. This book explains how Detect BS checks whether a model really learned something useful or merely found an easy way to look smart. The goal is not to make you do advanced algebra. The goal is to make the ideas behind the algebra understandable, whatever your background.

The Whole Journey on One Page

Before any of the details, here is everything the software does to your data, in order. Every question it asks is written out, and beside each question is what it actually DOES about the answer. The rest of this book is one section per idea in this picture.

What happens to your data, step by step

Every file goes through all of this, in this order. Follow the numbers: 1 to 6, then 7 to 10, then 11 to 16.
Blue = a question the software asks. Green ✓ = fine, keep going. Orange ! = it found a problem, and this is what it did about it.

PART ONE · GETTING YOUR DATA READY

1 Open the file and count it
How many rows? Which columns are clues, and which one is the answer?

2 Is the answer a number, or a name?
- ✓ A number, like sales? Use a straight-line output.
- ! A name, like yes or no? Use a squashed 0-to-1 output.

3 Are the rows in time order?
- ✓ Yes: test only on the newest rows.
- ! Too few rows for that? One 80/20 split instead.

4 Cut the rows: past to learn, future to test
They are put away, and not looked at again until scoring.

5 Measure the rulers using training rows ONLY
If test rows helped set the ruler, the model already knows something about its own test.

6 Did scaling make two columns identical?
- ✓ No: every column still says something different.
- ! Yes: say so BEFORE any score, and name the fix — put every column on one shared ruler.

PART TWO · BUILDING IT WITHOUT CHEATING

7 How many knobs can this much data support?
- ✓ Within budget: this shape may try.
- ! Too big: thrown out before training, so it cannot win by memorising the answers.

8 Train each allowed shape the same way
The same practice rounds for every shape, so the comparison is fair.

9 While training: is anything going wrong?
- ✓ No: keep going, and bookmark the best so far.
- ! A part stuck at 0 or 1? Reset that part. A step far too big? Trim it.

10 Put back the bookmarked best model
Not the last one, which is usually worse.

Why we bookmark instead of the last try

PART THREE · IS THE ANSWER REAL?

11 Does it beat SCRAMBLED answers?
- ✓ Yes: it beat a broken copy of itself.
- ✗ No: STOP. The score came from the method, not from your data. Nothing was learned.

12 Did an easier test make it look better?
- ✓ Small gap: this kind of cheating was absent.
- ! Better by over 15%: report a leak. It was filling in gaps, not predicting ahead.

13 Which clues is it really using?
- ✓ Score drops when a clue is scrambled: it used it.
- ! Nothing changes: it was never really using it.

14 Do five models trained separately agree?
- ✓ They agree: the data decided it, not luck.
- ! They scatter: they cannot all be right.

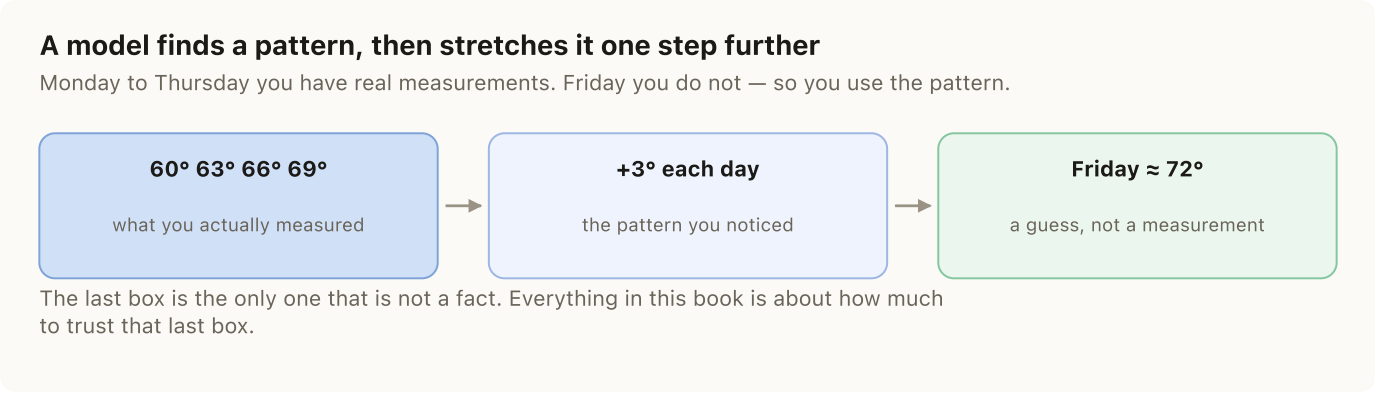
15 Does it beat plain guessing?
- ✓ Beats the average AND the last known value.
- ! Loses to either: it is not earning its place.

16 The report card: every answer above is written down together — and anything it failed is written FIRST, where it cannot be missed.

Passing all sixteen still does not prove it will be right about tomorrow.

Figure 1 — Follow the numbers 1 to 6 down the first column, then 7 to 10, then 11 to 16. Green means "fine, keep going". Orange means the software found a problem and this is how it handled it.

1. What Is a Model?



A model is a rule-making machine. It studies examples, notices patterns, and uses those patterns to make a guess about something it has not seen yet. Imagine the temperature is 60°F on Monday, 63°F on Tuesday, 66°F on Wednesday, and 69°F on Thursday. You might guess Friday will be about 72°F. You just used a model: you found a pattern and extended it.

A computer model does the same thing, but it can look at many clues at once. If we want to predict ice-cream sales, the inputs might be temperature, rain, day of the week, school schedule, and yesterday's sales. The answer we want - tomorrow's sales - is the target. Training means showing the model examples so it can adjust itself. Detect BS is interested not only in whether a model can make a prediction, but whether the evidence says we should trust that prediction.

> **Simply put:** A model finds patterns in examples and uses them to make guesses.

2. A Neural Network Is a Giant Adjustable Math Machine

The knobs decide how much each clue counts

Predicting ice-cream sales. Training turns these dials until the guesses stop being wrong so often.

hot weather turned up high | yesterday's sales turned up | day of the week turned a little | rain turned down hard | colour of your shoes left at nothing

Nobody sets these by hand. The network is a machine full of dials, and learning means turning them.

A neural network sounds mysterious, but you can picture it as a machine filled with adjustable math knobs. The knobs are called weights. A weight tells the network how strongly one clue should matter. If we are predicting ice-cream sales, hot weather might get a strong positive weight, rain might get a strong negative weight, and shoe color should probably get almost no useful weight.

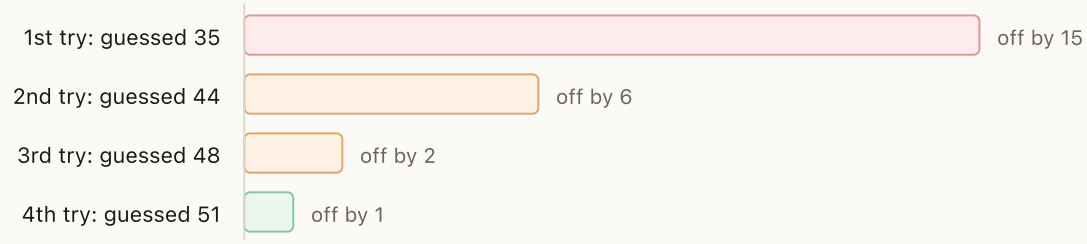
The network passes numbers through layers. Each layer combines what came before and passes a new set of numbers forward. The final layer gives the prediction. Detect BS uses small, inspectable networks on purpose. The goal is not to make the biggest possible AI. The goal is to make every important calculation understandable enough that another person can check it.

> **Simply put:** A neural network is a big adjustable math machine.

3. Learning Means Making Mistakes and Correcting Them

Learning is a shrinking mistake

The right answer is 50 ice creams. Watch the size of the error as it practises.



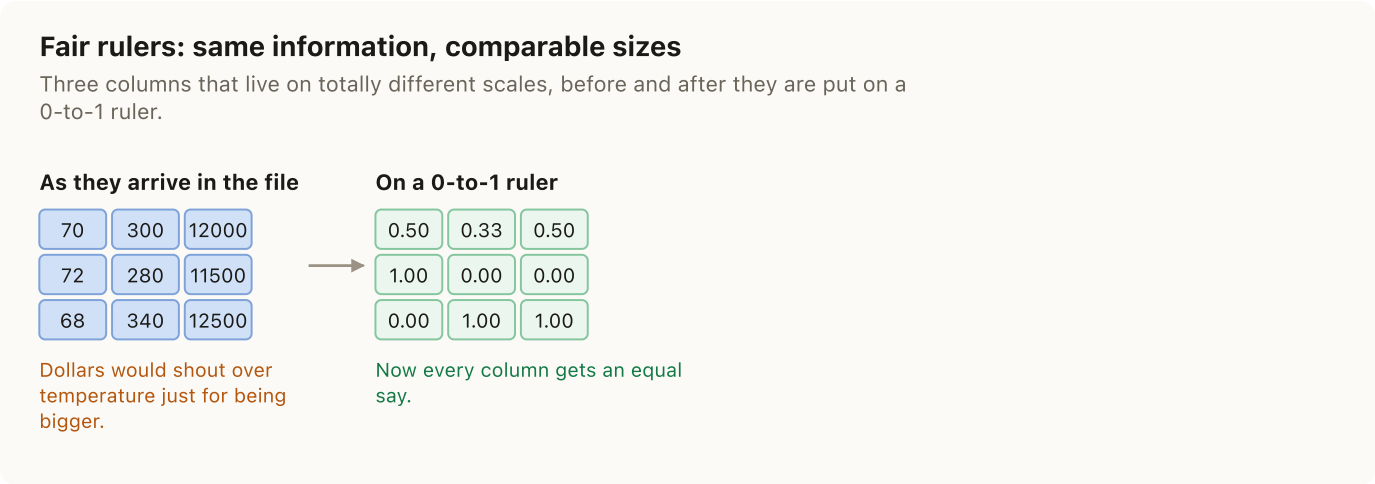
The mistake is the teacher. Without measuring how wrong it was, there is nothing to correct.

Neural networks learn by being wrong and then making small corrections. Suppose the correct answer is 50 ice creams, but the model predicts 35. The error is 15. The model changes its weights and tries again. Maybe it predicts 44, then 48, then 51. Each attempt gives information about how the weights should move.

Detect BS uses a method called stochastic gradient descent. That name sounds difficult, but the idea is simple: after each example, move the math knobs a little in a direction that should reduce the mistake. It is like practicing basketball. A shot goes left, so you correct to the right. A shot is short, so you use a little more force. Learning is repeated prediction, measurement, and correction.

> **Simply put:** Learning means making a guess, measuring the mistake, and correcting it.

4. Why the Numbers Need to Be Put on Fair Rulers

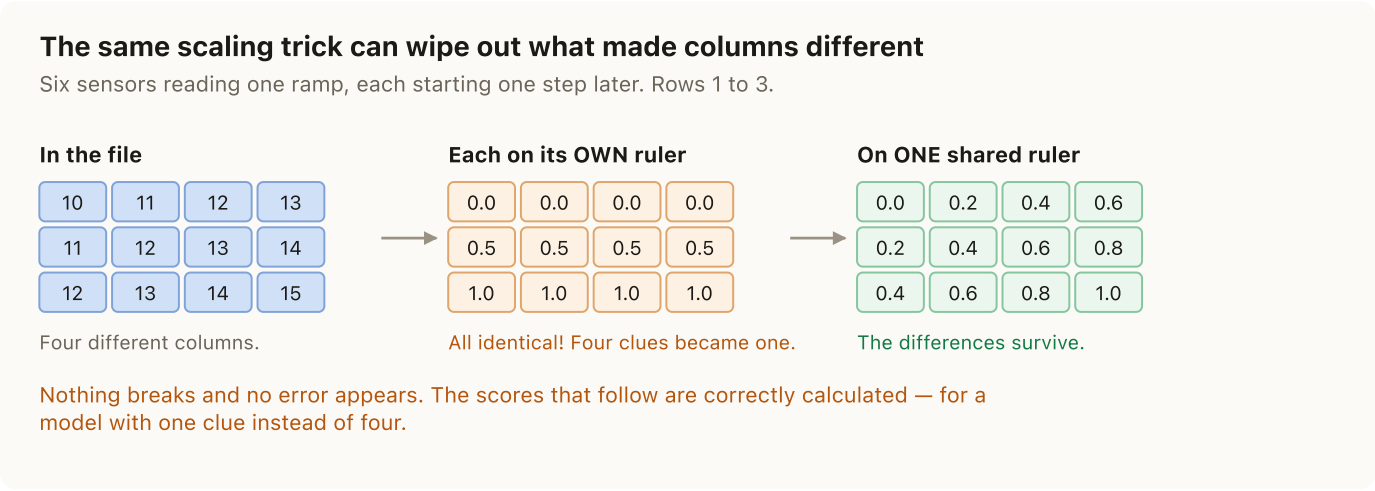


A dataset can mix numbers that live on very different scales. One column might be temperature around 70. Another might be customers around 300. Another might be dollars around 12,000. A neural network often learns more smoothly when the columns are placed on comparable rulers.

Detect BS scales each ordinary input column using its own minimum and maximum. If children's heights in the training data range from 40 to 60 inches, 40 can be mapped near 0, 50 near 0.5, and 60 near 1. The real-world meaning has not changed. The computer is simply using a friendlier internal ruler. The important rule is that the ruler must be learned from the training data only, not from future test data.

> **Simply put:** Scaling puts different kinds of numbers on friendlier rulers.

5. But Scaling Can Accidentally Destroy Information

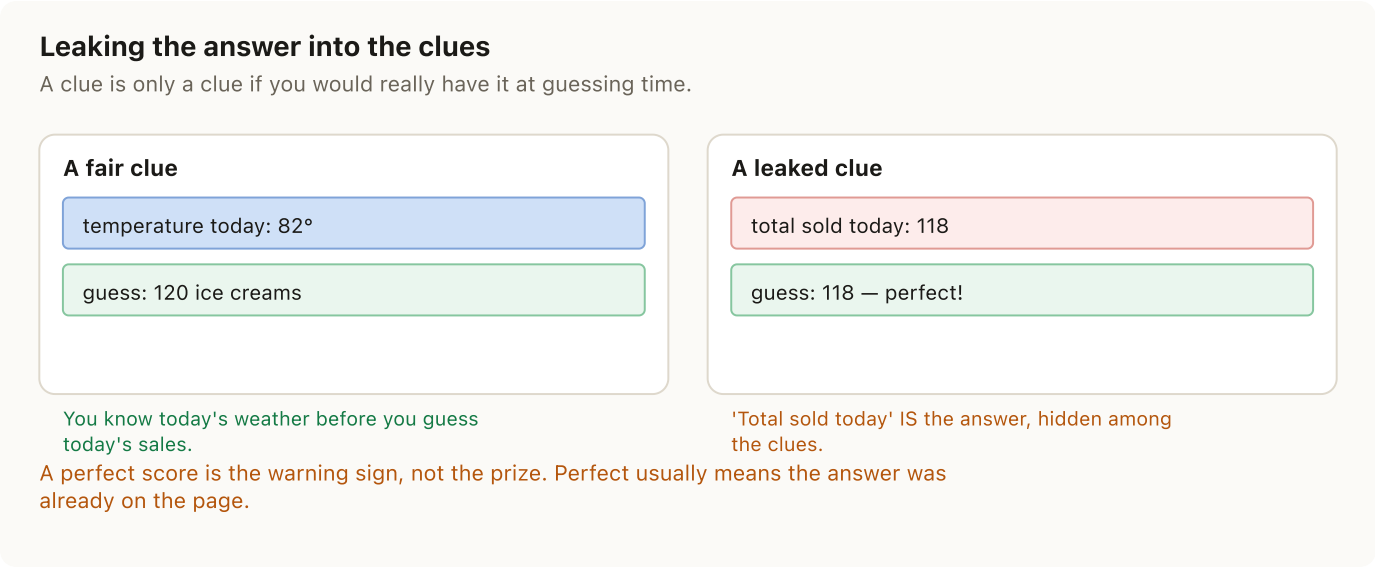


Scaling can sometimes erase differences that matter. Imagine six sensor columns that are really the same ramp shifted by one step: one starts at 10, another at 11, another at 12, and so on. If each column gets its own separate ruler, the shift can disappear. Six different columns may become six identical columns after scaling.

That is a serious problem because the model thinks it has six clues, but it really receives one repeated clue. Version 1.2.0 of Detect BS added a check for exactly this kind of collapse. It compares the columns before and after scaling and warns if distinct inputs become identical. Version 1.2.1 then fixed the repair itself. The warning named the setting that puts every column back on one shared ruler, but switching that setting on did not change anything you could see until the file was loaded again, so the cure looked broken even though the advice was right. Advice you cannot act on is worse than silence, because you stop believing the next warning too. The big lesson is that a calculation can be perfectly correct while describing damaged information.

> **Simply put:** A preprocessing step can accidentally erase information.

6. The Sneakiest Problem: Accidentally Showing the Computer the Answer

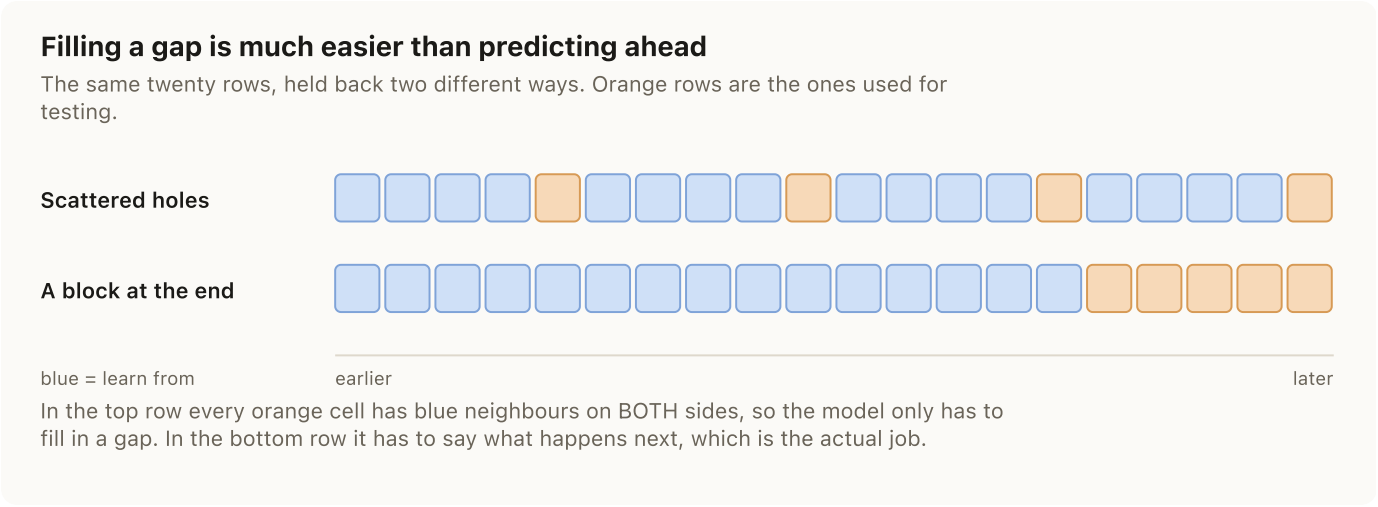


Imagine your teacher gives you tomorrow's spelling test today, including the answers. If you score 100% tomorrow, the score does not prove you are an amazing speller. You were allowed to peek. Machine-learning tests can have the same problem. It is called leakage.

For time-ordered data, Detect BS uses walk-forward testing. The model trains on the past and tests on the future. Then the training window can grow and the next future block is tested. The key rule is simple: the newest training time must still be earlier than the oldest testing time. Future information must not sneak backward into the model.

> **Simply put:** A fair future test never lets the model peek at tomorrow.

7. Why Randomly Mixing Past and Future Can Fool Us

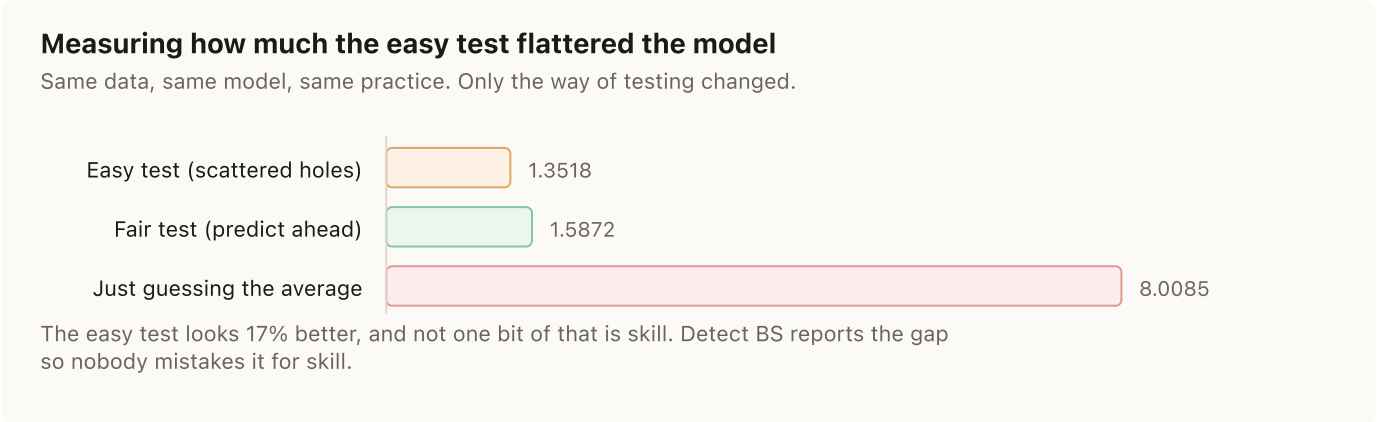


Suppose a child's height is 45 inches at age 6, 48 at age 7, 51 at age 8, 54 at age 9, and 57 at age 10. If we hide age 8 but let the model see ages 7 and 9, guessing 51 is easy. The missing point sits between two known points. That is interpolation.

Forecasting is harder. If the model sees ages 6, 7, and 8, then it must predict age 9 without already knowing age 10. When time data is randomly mixed, the model may get an unfairly easy job. It can fill gaps between known neighbors instead of facing the real future. Detect BS treats those as different problems.

> **Simply put:** Filling a gap between known points is easier than forecasting the future.

8. Detect BS Measures How Much the Easy Test Helped



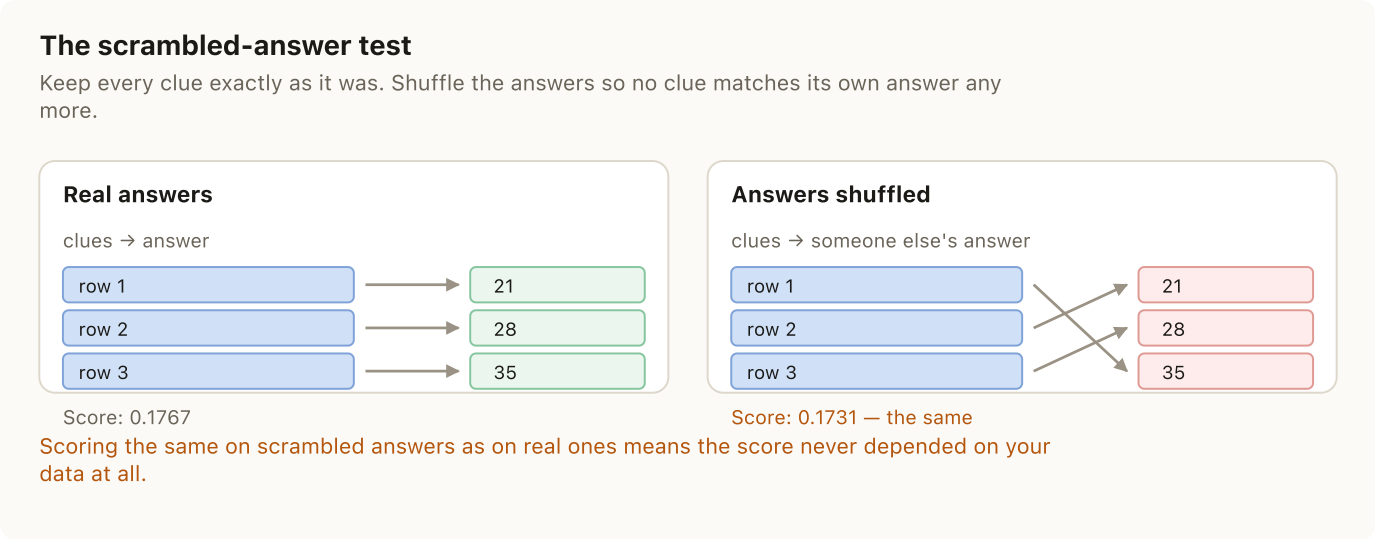
Detect BS can compare an easy interleaved test with a harder walk-forward test. Suppose the easy test has an error of 10, but the fair future test has an error of 20. The easy method made the model look twice as good.

The software reports a leak gap. If the easy split looks much better than the future-facing split, that is evidence that the test method flattered the model. The document uses a gap above 15% as a leak warning. A smaller gap is reported more cautiously. This is good scientific behavior: a warning should say exactly what the evidence supports, not more.

> **Simply put:** Compare testing methods to see whether one made the model look unfairly good.

---

9. The Scrambled-Answer Test



One of the strongest checks is the negative control. Suppose study hours and scores have a real pattern: 1 hour -> 55, 2 -> 63, 3 -> 72, 4 -> 81, 5 -> 91. Now scramble the scores so the pairings are nonsense. The dataset still has the same number of rows and the same kinds of values, but the real relationship is gone.

Train the same model on the scrambled targets. If it performs about as well as it did on the real targets, something is seriously wrong. The model's good score was probably coming from the procedure, not from a real learnable relationship.

> **Simply put:** If scrambled answers work as well as real answers, the model did not prove useful learning.

10. Why Passing the Scrambled Test Does NOT Prove the Model Is Good

Passing the negative control is important, but it is not a magic certificate. Think of checking a bicycle. If the brakes work, that is good news, but it does not prove the tires, handlebars, chain, and frame are all safe.

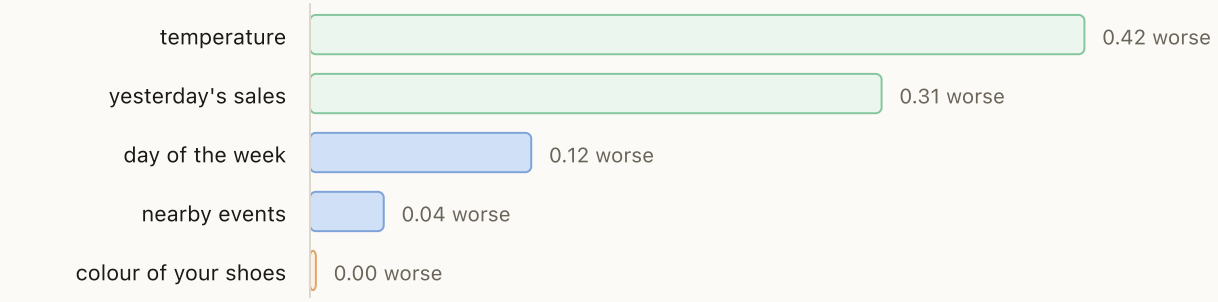
The negative control works the same way. If a model fails it, the failure is very meaningful. But if it passes, we only know it beat one deliberately broken version of the problem. Other problems may still exist. Detect BS repeatedly makes this distinction between 'this check passed' and 'the whole model is proven.'

> **Simply put:** Passing one check never proves everything is safe.

11. Which Information Is the Computer Actually Using?

Break one clue at a time to see which ones matter

Scramble a single column, score again, and measure how much worse it got. Bigger bar = the model was relying on it more.



Shoe colour costs nothing to destroy, so it was never being used — whatever its dial happened to be set to.

Permutation importance asks what clues the model actually depends on. Imagine an ice-cream model with temperature, rain, weekday, nearby people, and the owner's shoe color. Shuffle the temperature values across test rows. If the model suddenly becomes much worse, temperature mattered. Shuffle shoe color. If almost nothing changes, shoe color was not doing useful work.

This test is better than simply staring at internal weights. A large weight can sit on a nearly constant input and accomplish almost nothing. Permutation importance measures what happens when the information is actually broken.

> **Simply put:** Break one input at a time to see what the model really uses.

12. 'Important to the Model' Does NOT Mean 'Causes the Result'

Used by the model ≠ causes the answer

Sunglasses sales predict ice-cream sales very well. Neither one causes the other.

A model told only about sunglasses will use sunglasses, and will look clever right up until a cloudy day with a heatwave.

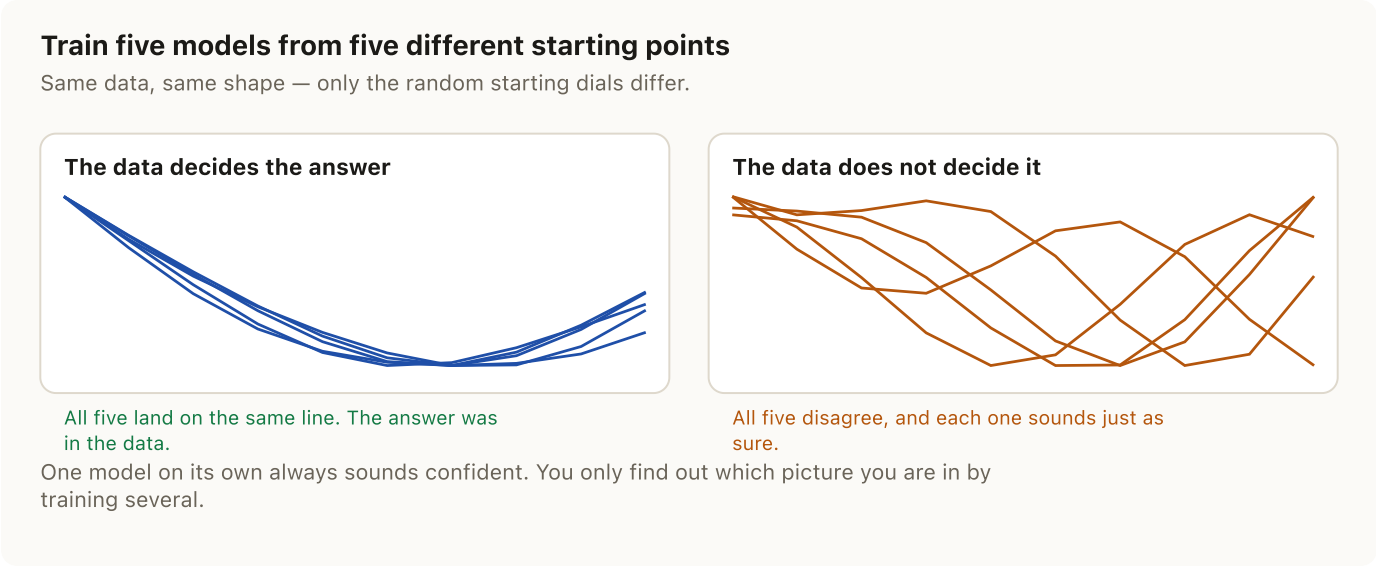
Suppose ice-cream sales rise on days when more people wear sandals. A model may correctly use sandals as a helpful clue. But sandals do not make people buy ice cream. Hot weather causes both more sandals and more ice-cream buying.

Permutation importance tells us what the model relied on. It does not tell us what causes what in the real world. A model can rely strongly on an accidental or misleading relationship. That is why the words 'important feature' should never automatically be changed into 'cause.'

> **Simply put:** What the model uses is not automatically what causes the result.

---

13. Ask Several Models Instead of Trusting One



Imagine five classmates guessing the number of jellybeans in a jar. If they say 399, 401, 402, 403, and 405, they strongly agree. If they say 120, 300, 425, 700, and 900, the answer is unstable.

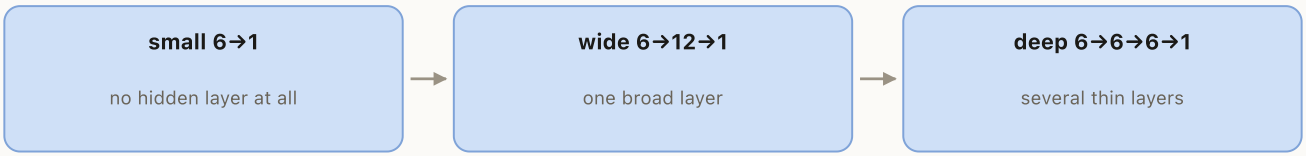
Detect BS trains several models and looks at their spread. When models trained on the same data give similar answers, the prediction is more stable. When they give wildly different answers, the data may not strongly determine one answer. This group of models is called an ensemble.

> **Simply put:** Several models can reveal whether an answer is stable.

14. The Models Can Also Be Built Differently

The five models can also be built differently on purpose

Not just different starting dials — genuinely different shapes, all trained on the same rows.



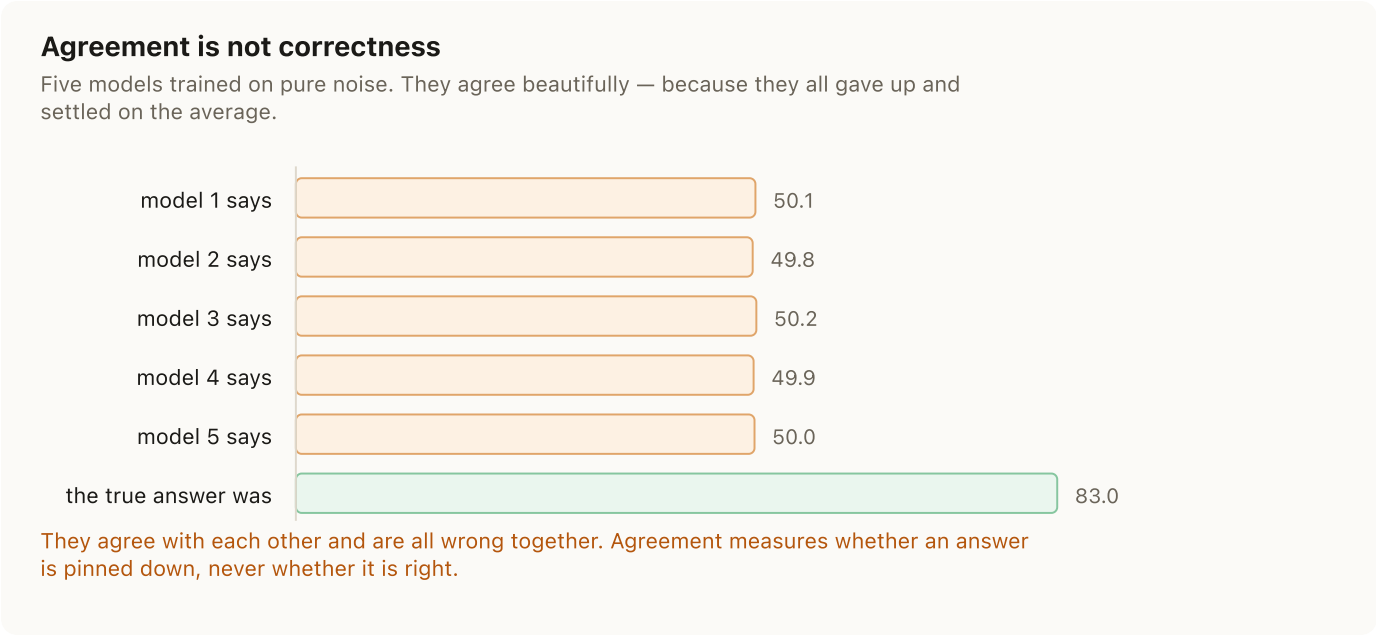
Agreeing from different starting points says the answer is not luck. Agreeing from different SHAPES says it does not depend on the particular model somebody chose — a stronger and rarer thing to check.

An ensemble does not have to contain identical network shapes. One model can be small, another wider, and another deeper. If several reasonable designs all arrive at similar answers, that is useful evidence that the answer is not just an accident of one architecture.

With identical shapes, spread mostly measures sensitivity to random starting points. With different shapes, spread can measure sensitivity to specification - the choice of model itself. Detect BS also checks whether the models fit comparably well, because a terrible model disagreeing with good models is not useful evidence of uncertainty.

> **Simply put:** Different reasonable model designs can test whether the answer depends on the design choice.

15. But Agreement Does NOT Mean Correctness

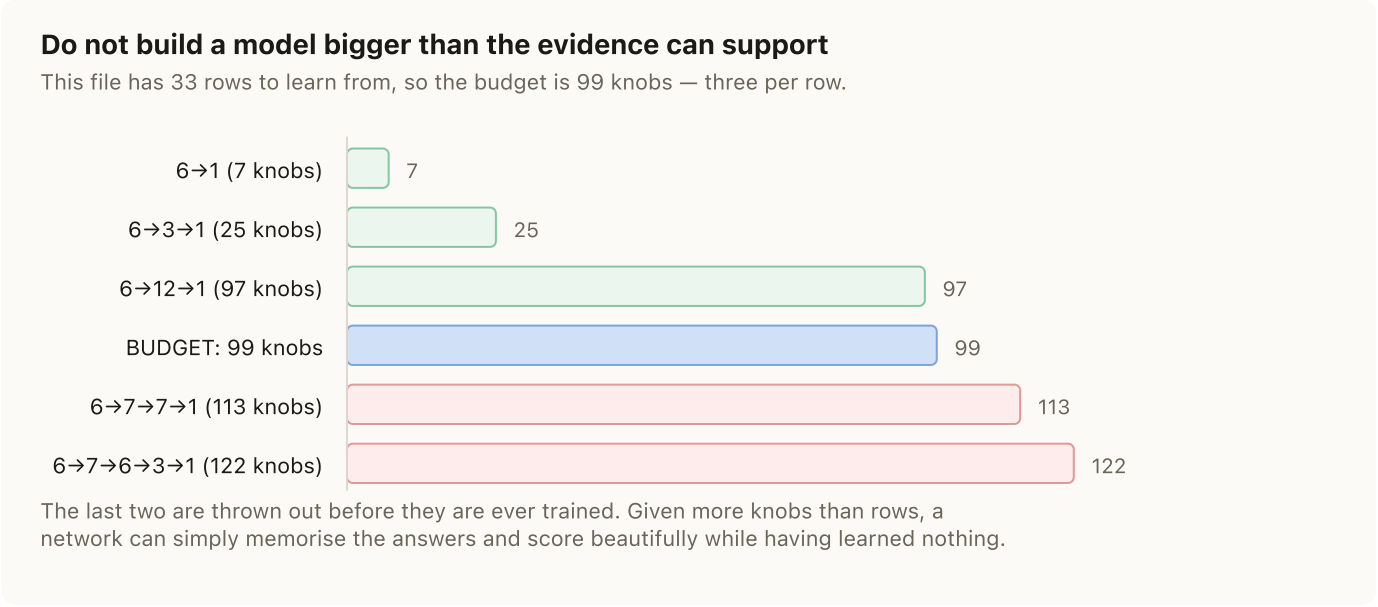


Five models can agree and still be wrong. Imagine asking five people how many dragons live on the Moon. They may all say zero. Agreement tells us their answers are stable, not that their reasoning proves anything.

Detect BS says ensemble spread measures determinacy, not accuracy. Even models trained on noise can agree because they may all settle near an average. Agreement is useful evidence about stability. It is never a substitute for checking whether the model is actually right on honest test data.

> **Simply put:** Agreement is not the same thing as correctness.

16. Do Not Let the Neural Network Become Bigger Than the Evidence

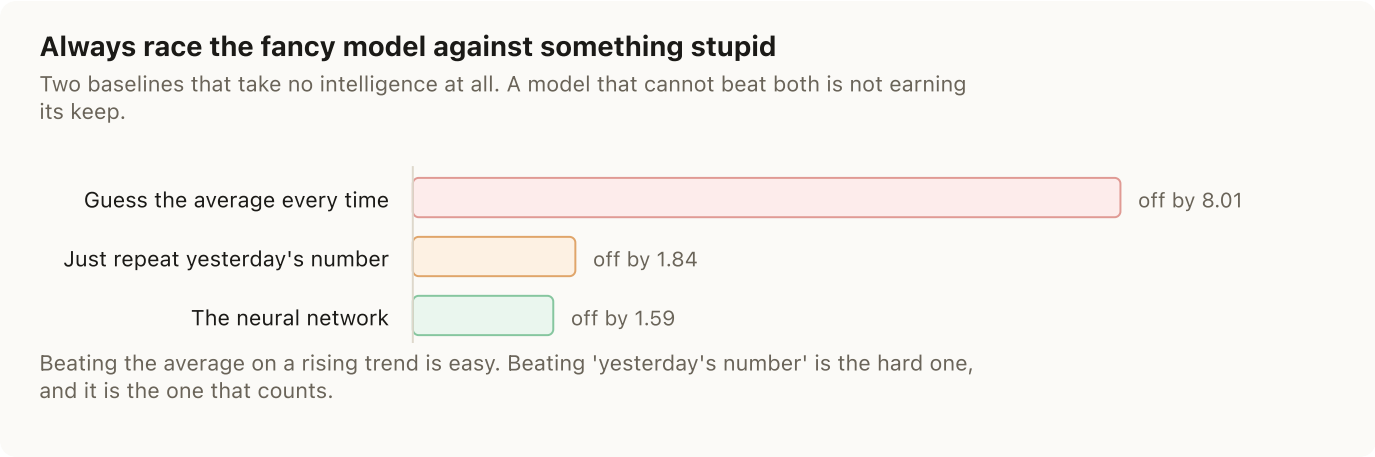


A very large model can memorize a small dataset. Imagine a student who memorizes that 7×8=56, 6×9=54, and 5×7=35 but does not understand multiplication. The student may score perfectly on those exact questions and fail on 8×9. That is like overfitting.

Detect BS sets a capacity budget. A candidate network with too many trainable parameters compared with the amount of training data can be rejected before training. The idea is simple: do not give the model thousands of adjustable knobs when you only have a tiny pile of evidence.

> **Simply put:** A model should not be much more complicated than the evidence can support.

17. Always Compare the Fancy Model With Something Stupid

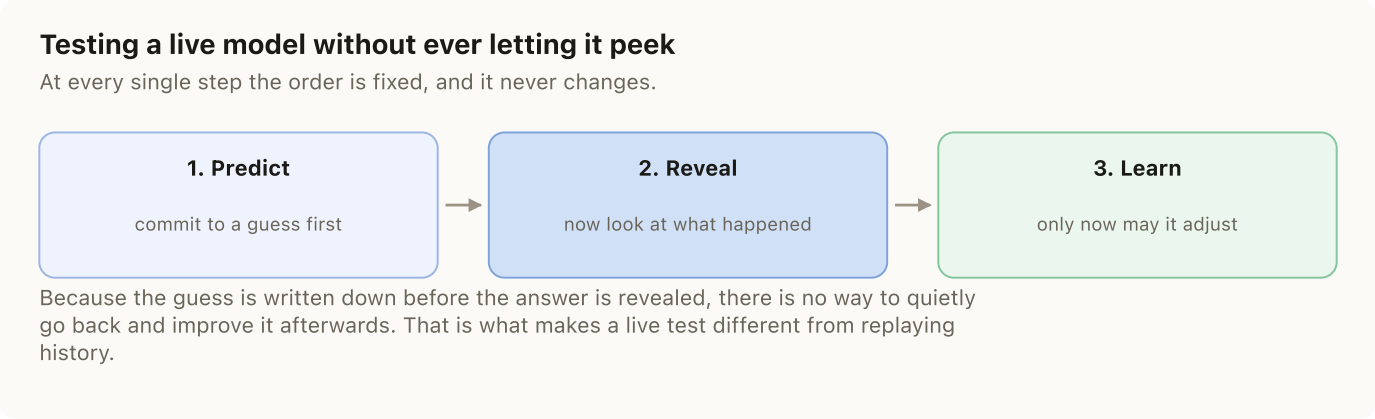


A complicated model needs to beat a simple baseline. Suppose a giant weather AI predicts tomorrow within 3 degrees on average. Then someone with no model at all simply says, “Tomorrow will be the same temperature as today,” and gets within 2 degrees. The simple rule wins.

For time series, Detect BS pays special attention to a no-change baseline. Beating the average value can be easy when a series is trending. Beating the last known value is harder and more meaningful. Fancy mathematics should earn its complexity by beating a simple rule.

> **Simply put:** A fancy model should beat a simple baseline.

18. Testing a Live Model Without Letting It Peek



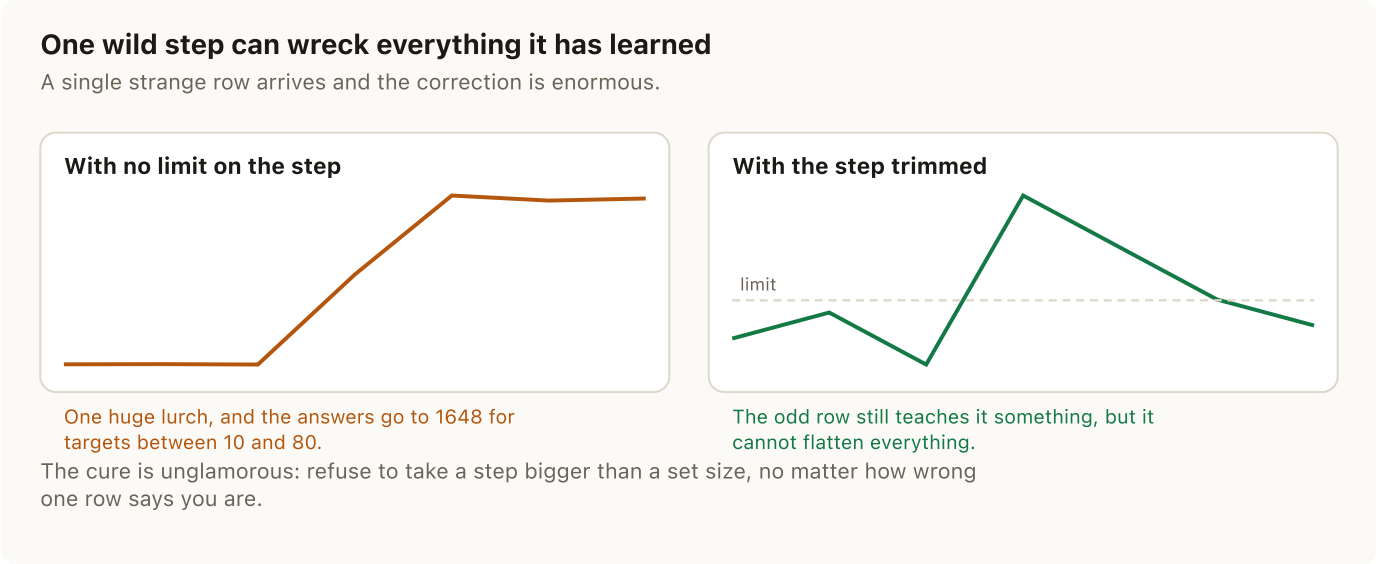
Streaming evaluation follows a strict order: predict first, then learn. On Monday the model predicts Tuesday. Tuesday arrives. Only then may the model learn from Tuesday before predicting Wednesday.

This is called prequential evaluation. Every prediction is made with only the information available at that time. It prevents the model from learning the answer and then pretending that it predicted it. That makes streaming evaluation feel much more like real use than a test that can be refitted after the fact.

> **Simply put:** In live testing, predict first and learn afterward.

---

19. Sometimes Learning Can Go Wild

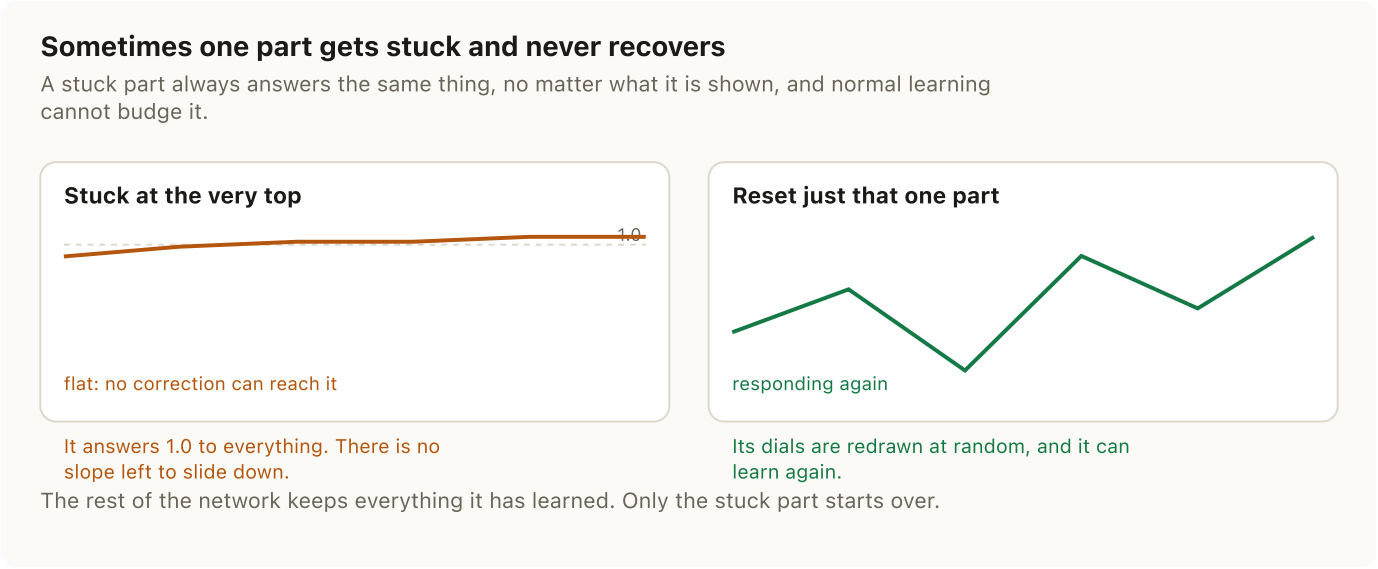


A neural network can make one enormous mistake and then overreact. The document gives an example where a prediction of 702 was made for a target of 17. A huge correction can shove internal weights so far that parts of the network stop learning.

Detect BS uses clipping and step caps. Think of bumpers on a bowling lane. The ball can still move and correct its direction, but one wild roll cannot fly completely out of the useful area. The goal is not to stop learning; it is to prevent one strange example from destroying the learner.

> **Simply put:** Limit giant learning steps so one strange example cannot wreck the model.

20. Sometimes Part of a Neural Network Can Get 'Stuck'

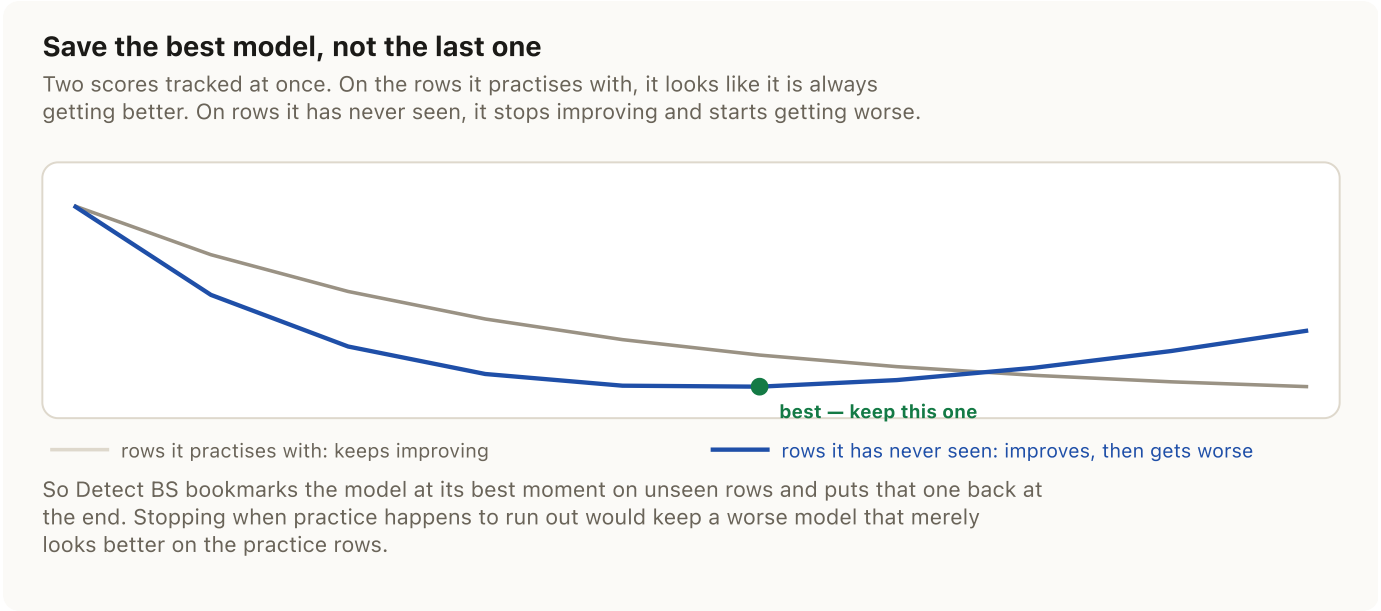


Some neural-network nodes use a sigmoid, which behaves like a dimmer switch. In the middle, small changes matter. At the extreme ends, the node becomes almost completely off or on, and its learning signal can become nearly zero. Then it may stay stuck.

Detect BS can reinitialize the incoming weights for a dead node. It is like restarting one frozen calculator instead of throwing away the entire classroom project. The rest of the network keeps its learned structure while the stuck part gets another chance to learn.

> **Simply put:** Sometimes one stuck part should be reset instead of restarting everything.

21. Save the Best Model, Not Just the Last Model

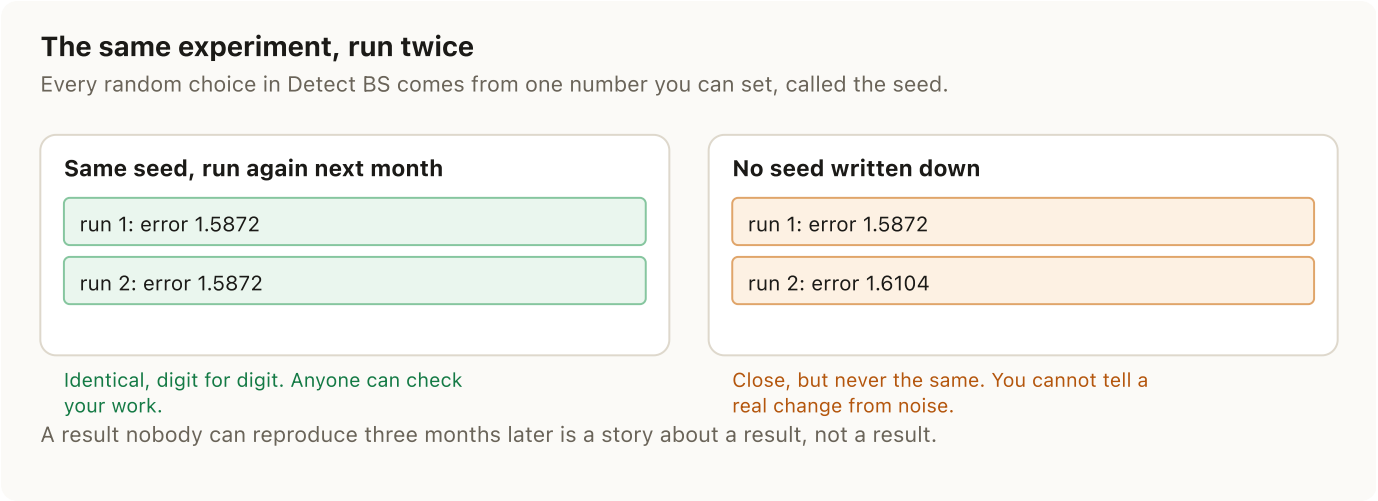


Training does not always improve forever. Imagine practicing basketball for six hours. At hour four you are sharp. At hour six you are tired and missing. The last performance is not automatically the best one.

Detect BS saves a snapshot when validation performance is best and restores that version at the end. This is like putting a bookmark on the strongest version of the model. It helps prevent the final training steps from erasing a better solution found earlier.

> **Simply put:** Keep the best validated version, not automatically the final version.

22. Repeating the Experiment Matters

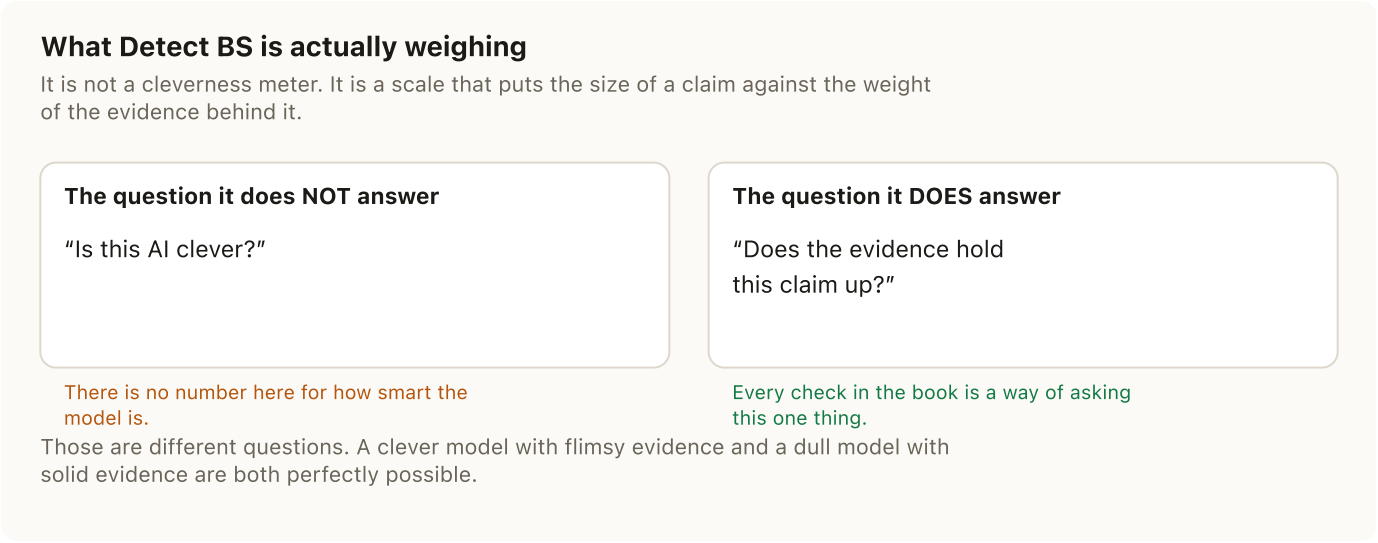


Science is stronger when another person can repeat an experiment and get the same result. Detect BS uses a user-settable random seed so that its random choices can be repeated. The seed controls things like starting weights, shuffled targets, input permutations, and ensemble variations.

The software can also save a trained engine. A result that cannot be reproduced later is much harder to treat as solid evidence. Reproducibility turns a one-time story into something another person can inspect and test.

> **Simply put:** A scientific result should be repeatable.

23. What Detect BS Is Really Testing

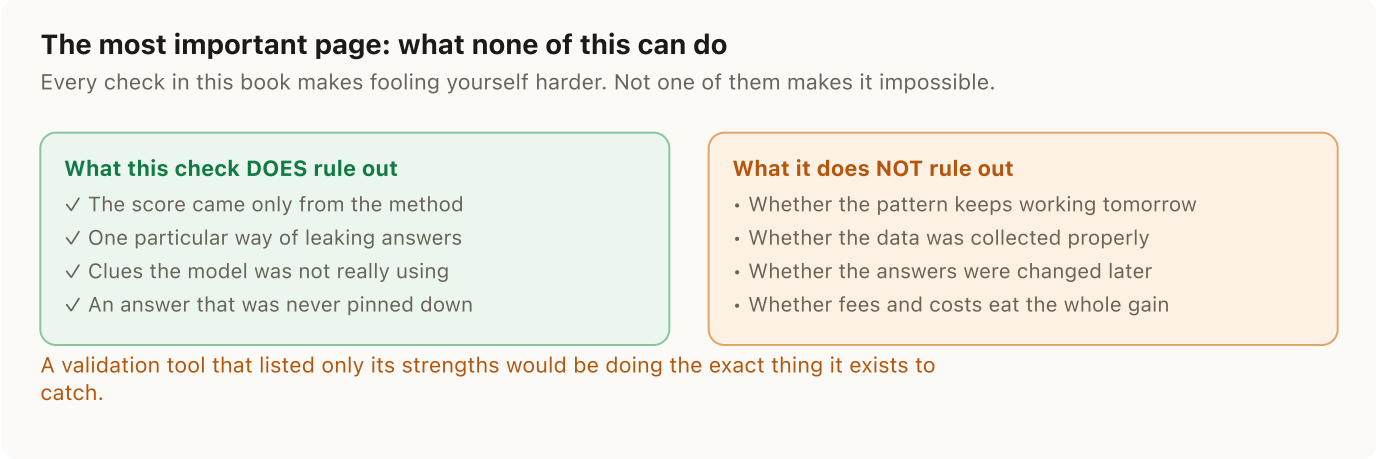


Detect BS does not ask only one question such as 'What is the accuracy?' It asks a whole family of questions. Did the model really face unseen data? Did the future leak into the past? Does it beat a simple baseline? Does it beat shuffled nonsense? Which clues does it use? Do several models agree? Is the model too complicated for the evidence? Can the result be repeated?

This collection of checks matters because a single score can hide many different failures. A good-looking number is only the beginning of the investigation.

> **Simply put:** Trust comes from many checks, not one score.

24. The Most Important Section Is About What Detect BS Cannot Do



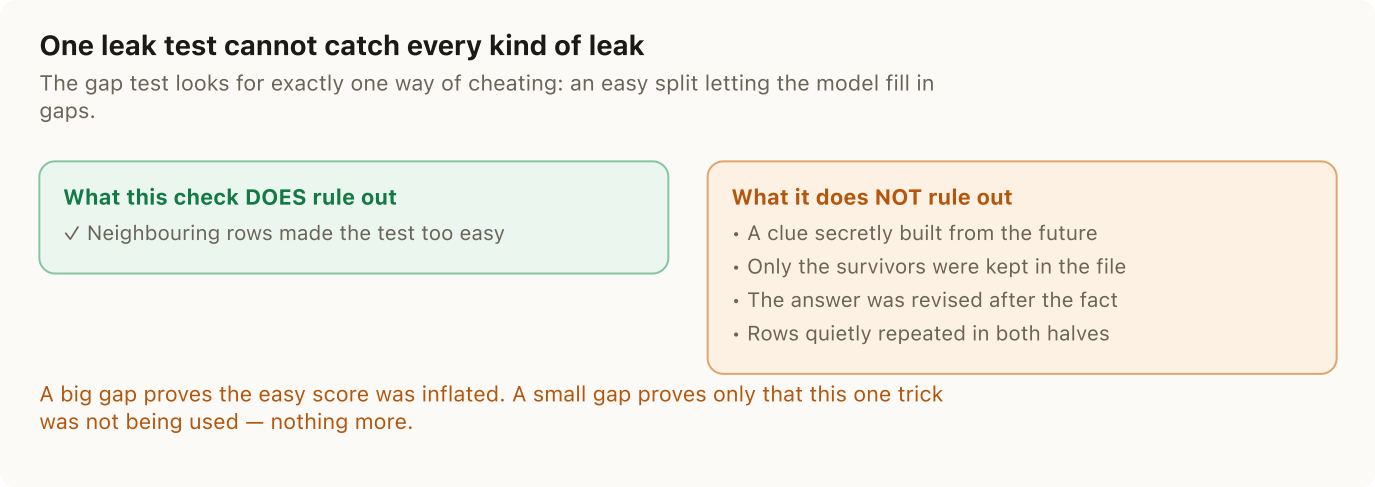
The document says its limitations are especially important. No validation method can prove that a relationship found in old data will continue forever. Conditions change. People change. Machines wear out. Markets change. Weather patterns shift. A model can pass every historical test and still fail tomorrow.

This is not a weakness unique to Detect BS. It is a basic limit of prediction. Good science says clearly what the evidence supports and what it cannot guarantee.

> **Simply put:** No historical test can guarantee the future.

---

25. A Leak Test Cannot Find Every Kind of Leak

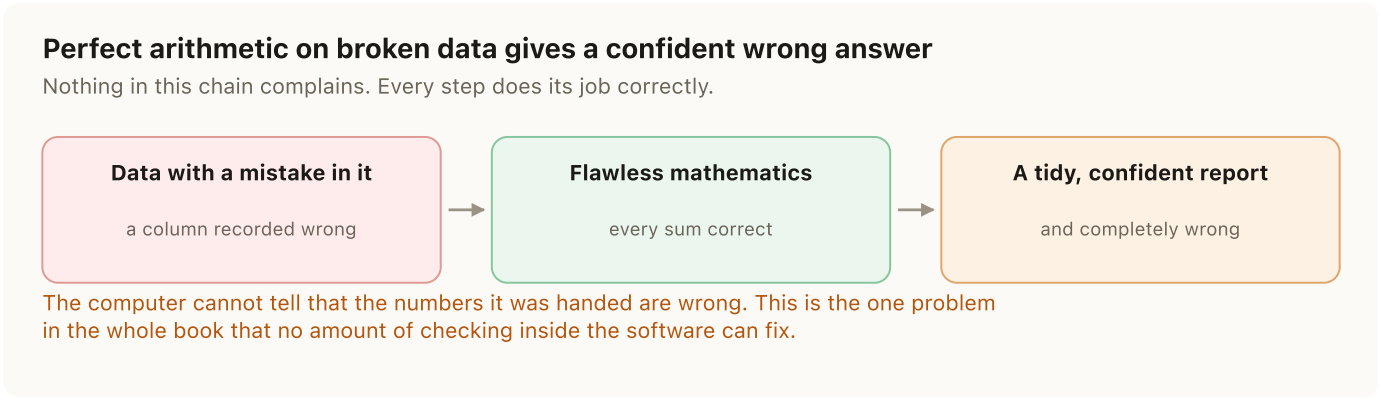


A leak check can test one kind of unfair peeking, but it cannot magically detect every possible leak. Perhaps the rows are split correctly, but one input column was calculated using tomorrow's answer. The time split looks perfect, yet the feature itself contains future information.

A small leak gap therefore means only that one particular leakage mechanism was not strongly detected. It does not prove that all leakage, bias, or data mistakes are absent. A good warning label names the exact thing that was checked.

> **Simply put:** A test can rule out one problem without ruling out every problem.

26. Bad Data Can Still Produce Beautiful Mathematics

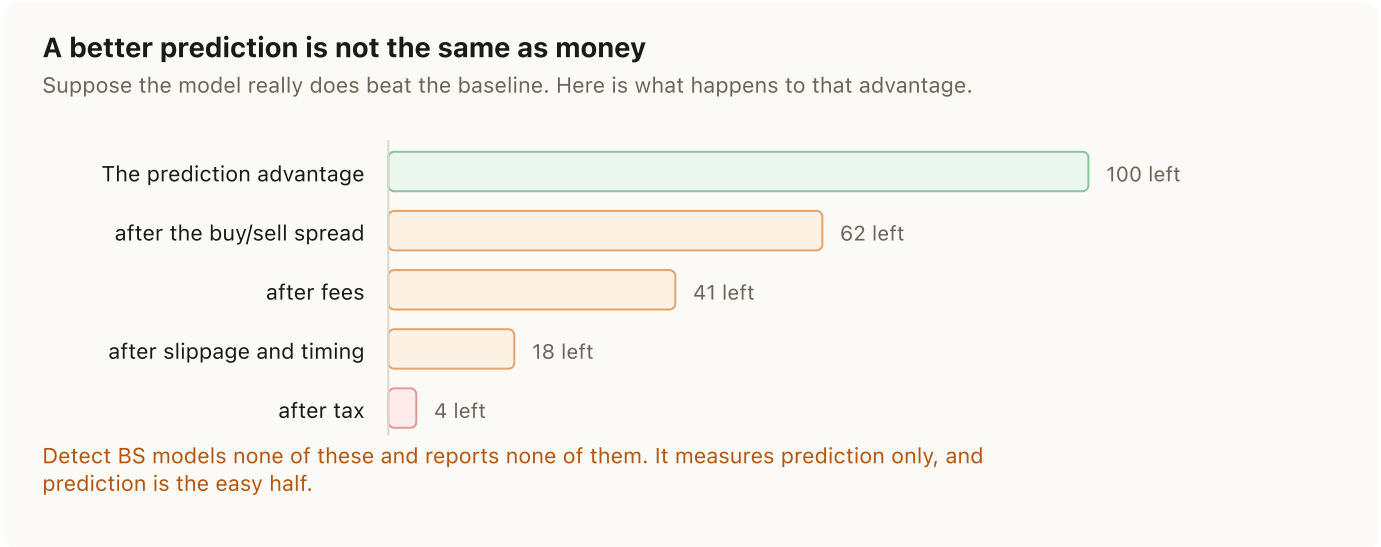


A computer can perform flawless mathematics on incorrect data. Suppose a survey asks how many pets children own. Emma says 2 but the computer records 20. Noah says 1 but it records 10. Ava says 3 but it records 30. The equations can be perfect while the conclusion is nonsense.

Detect BS cannot always discover incomplete, mislabeled, biased, revised, or secretly future-informed data by looking at the file alone. This is the old lesson 'garbage in, garbage out.' Correct arithmetic does not rescue bad evidence.

> **Simply put:** Perfect math on bad data still gives a bad answer.

27. A Better Prediction Does Not Automatically Mean Money



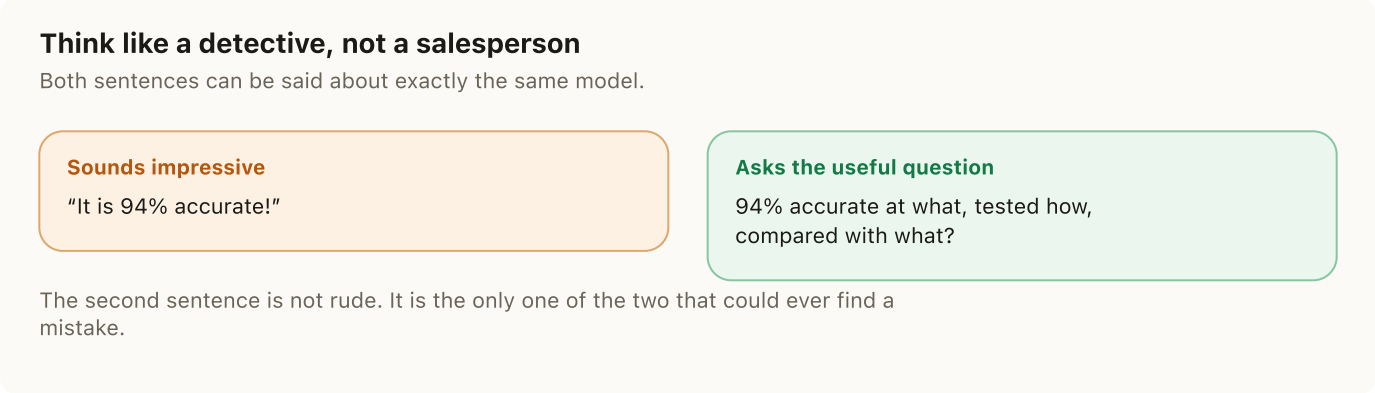
A model might beat a simple price-prediction baseline and still fail to make money in the real world. Buying and selling can include fees, spreads, taxes, delays, limited liquidity, and prices that move before an order is completed. Detect BS does not model all of those things.

So 'better prediction' and 'profitable strategy' are different claims. This is a great example of why a model's conclusion should never be stretched beyond what the test actually measured.

> **Simply put:** Better prediction is not automatically the same thing as making money.

---

28. Think Like a Detective, Not a Salesperson

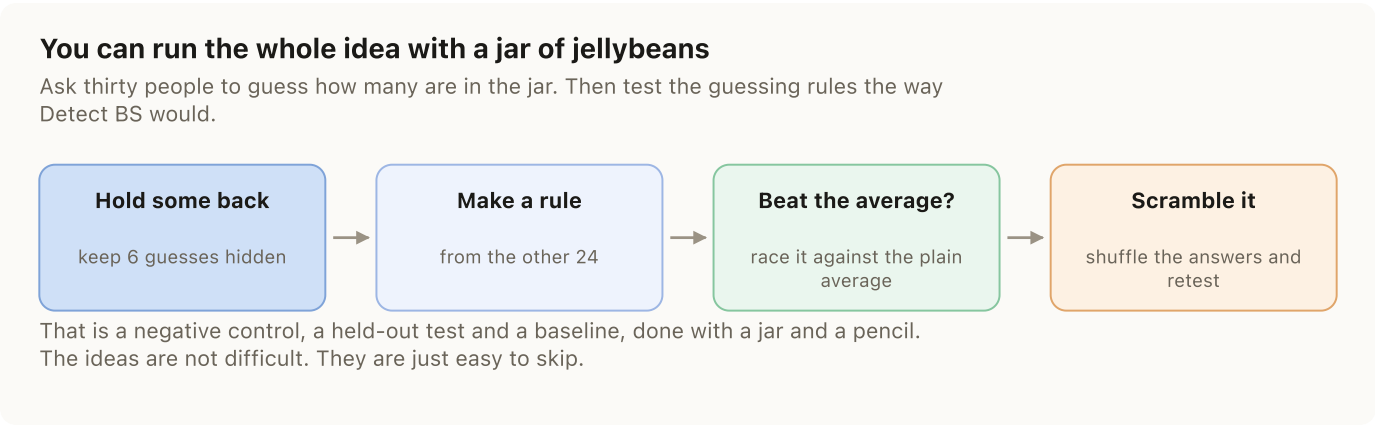


A salesperson may say, 'My model is 95% accurate!' A detective asks how that 95% was measured. Was the test fair? Was the future hidden? Did shuffled data fail? What baseline was beaten? Which clues mattered? Did several models agree? What are the limitations?

Detect BS encourages the detective's attitude. The goal is not to make a model sound impressive. The goal is to make an impressive claim survive attempts to break it.

> **Simply put:** Ask how the impressive number was earned.

29. A Kitchen-Table Detect BS Experiment

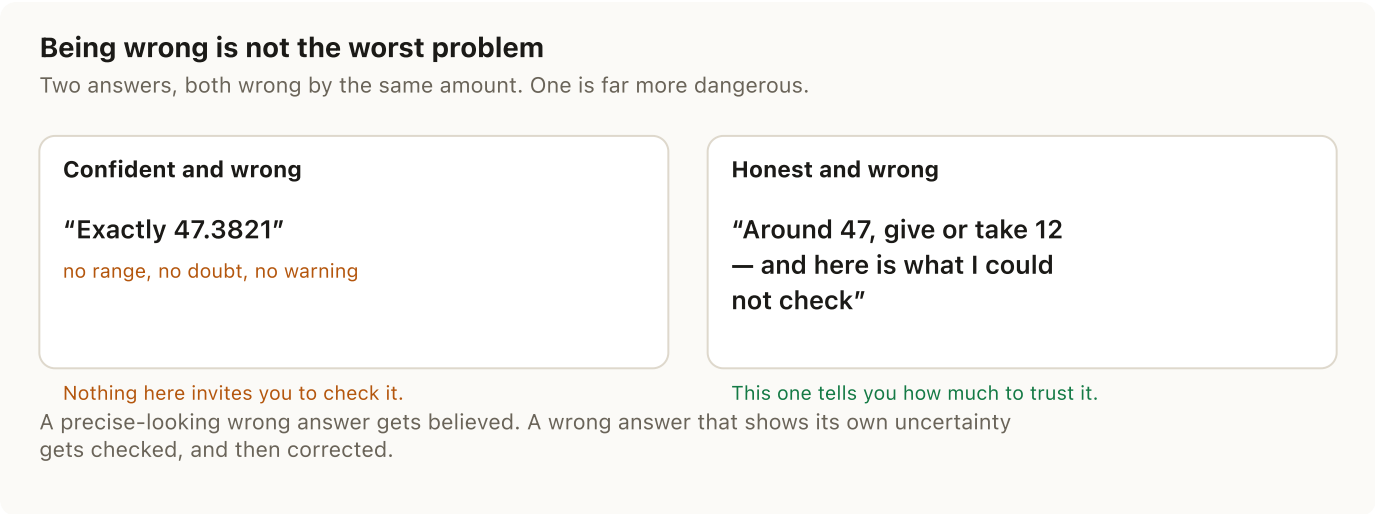


Imagine predicting jellybeans in jars from jar height, width, bean size, and weight. First collect many example jars. Then train a model. Before celebrating, ask: were test answers kept secret? Did the model beat a simple guess based only on jar weight? What happens if the jellybean counts are scrambled? What happens if jar width is scrambled? Do several models agree? Is the model much too large for only 100 jars? Can someone repeat the experiment?

That small classroom experiment contains almost the entire Detect BS philosophy.

> **Simply put:** Even a jellybean experiment can teach serious model validation.

30. The Biggest Lesson: Being Wrong Is Not the Worst Problem



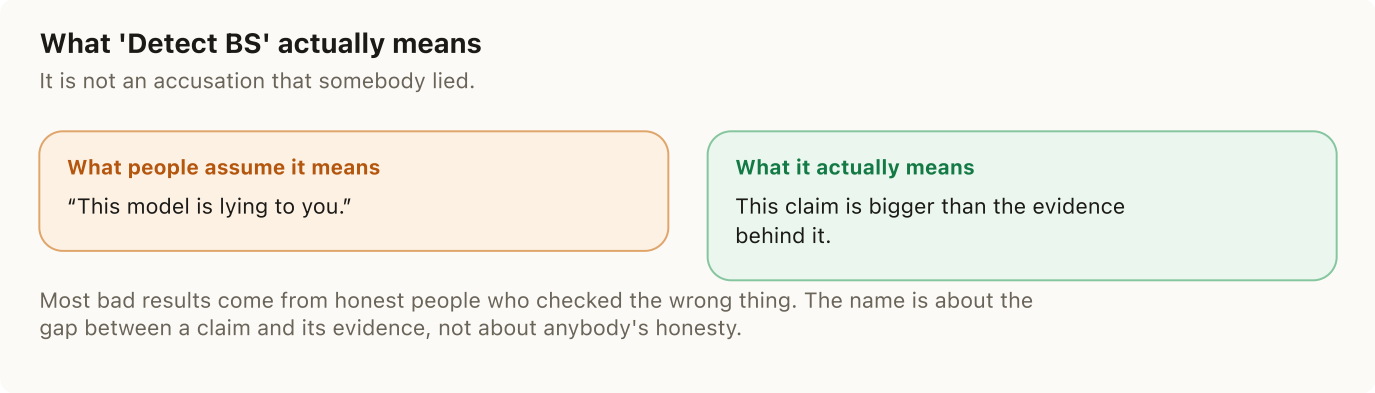
A model that says 'I don't know' can still be useful. A model that makes an obvious mistake can be fixed. A more dangerous model is one that produces a beautiful, confident number for the wrong reason.

A score such as 98.7000% looks scientific because it has decimal places. But the test could still be leaky, the dataset could be tiny, the scaling could have erased information, shuffled nonsense could score just as well, or a simple baseline could win. Decimal places measure precision of reporting, not truth.

> **Simply put:** A confident, precise-looking wrong answer can be more dangerous than an obvious mistake.

---

31. What 'Detect BS' Really Means

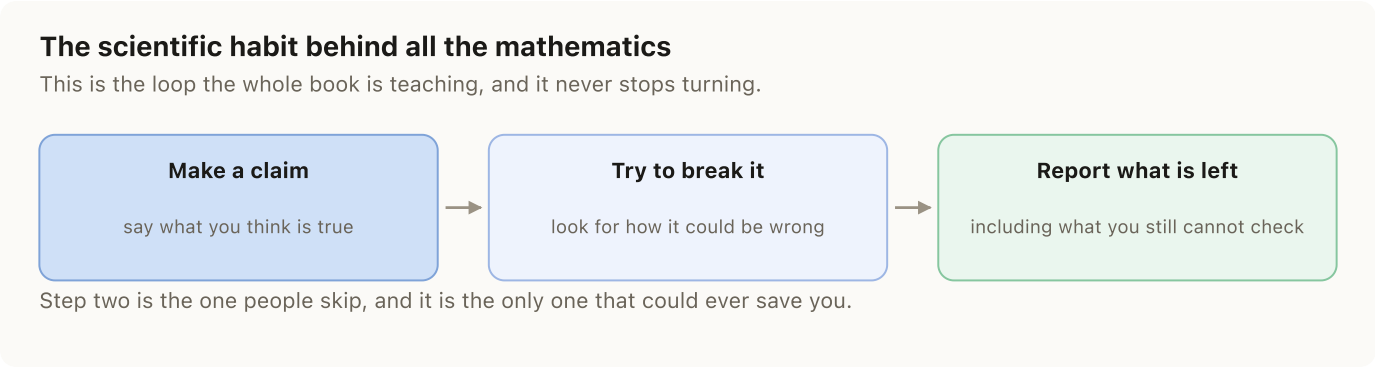


The title is funny, but the problem is serious. BS does not have to mean somebody is lying. People can fool themselves. Programmers want their model to work. Scientists want their experiment to succeed. Businesses want good news. Those hopes can quietly influence how a test is designed.

Detect BS uses mathematical checks to make self-deception harder. It cannot make self-deception impossible, but it forces the model's claims to face more than one kind of challenge.

> **Simply put:** Self-deception is a real risk, even when nobody is lying.

32. The Scientific Habit Behind the Mathematics

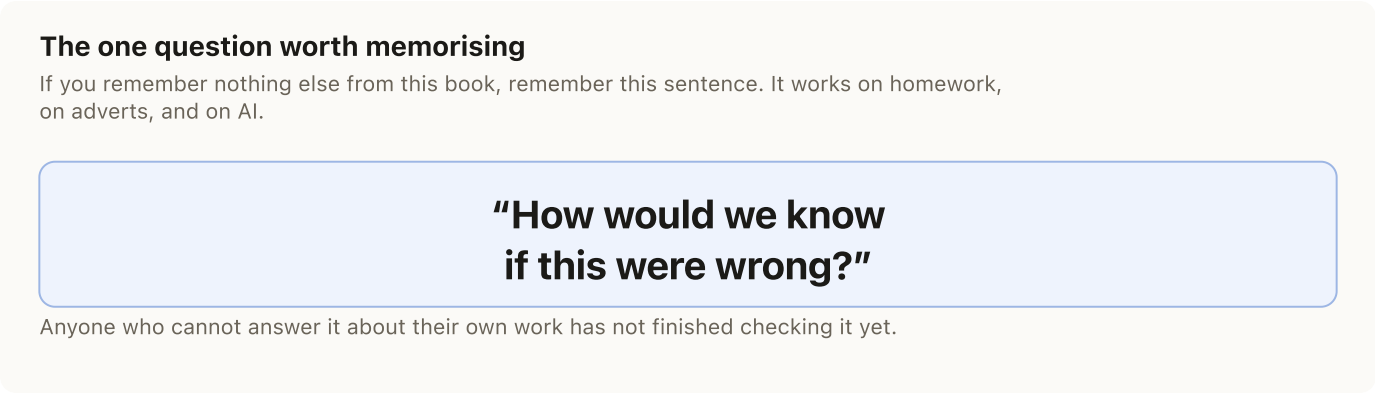


Under all the equations is a simple habit: do not stop at 'it worked.' Ask why it worked, how it was tested, whether the test matched real use, whether future information leaked in, whether nonsense data also looks good, whether a simple baseline wins, whether other models agree, and what remains unknown.

These habits are useful far beyond neural networks. They are basic habits of careful science, engineering, and everyday reasoning.

> **Simply put:** Good science asks how a result could be wrong.

33. The Ultimate Detect BS Question

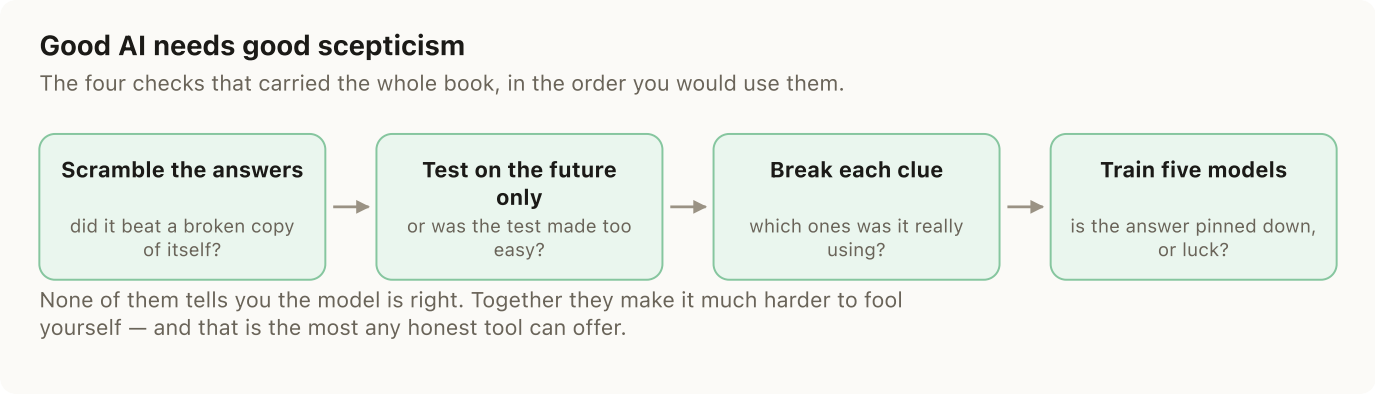


If someone says, 'My AI can predict the future,' do not ask only for its accuracy. Ask whether it was tested on information it had never seen. Ask whether the future was kept out of training. Ask what happens when the answers are scrambled. Ask whether it beats a simple baseline. Ask which inputs it uses. Ask whether several models agree. Ask whether reasonable model designs disagree. Ask whether the experiment can be repeated.

Then ask the most important question: What does your experiment NOT prove? A person who can answer that clearly is giving you much better evidence than someone who only points to an impressive score.

> **Simply put:** Always ask what the experiment does not prove.

Conclusion: Good AI Needs Good Skepticism



Artificial intelligence can find patterns that people miss, but powerful tools need powerful tests. A calculator can give an exact answer to the wrong equation. A neural network can find a pattern that disappears tomorrow. A model can score beautifully because it saw part of the answer. Five models can agree and still be wrong. An input can be important for an accidental reason. Perfect mathematics can run on terrible data.

Detect BS turns skepticism into a set of repeatable checks. It asks models to face the future honestly, compete against simple baselines, beat shuffled nonsense, reveal which clues they use, show whether several reasonable models agree, stay within a sensible complexity budget, and make experiments reproducible. Most importantly, it states what none of those checks can prove.

The best scientific sentence is not 'My answer must be right.' It is: 'Here is my answer, here is my evidence, here is how I tried to prove myself wrong, and here is what I still do not know.'

> **Simply put:** Good AI needs good skepticism.

Source and Reading Note

This illustrated guide is a plain-language adaptation of Detect BS - The Mathematics, Version 1.2.1, dated 16 August 2026. It states the explanations in plain language while preserving the document's main ideas: scaling, neural-network training, capacity control, walk-forward validation, leakage checks, negative controls, permutation importance, ensemble spread, baselines, streaming evaluation, reproducibility, and explicit limitations.

The original mathematics document is the authority for exact equations, parameter defaults, thresholds, and implementation details.