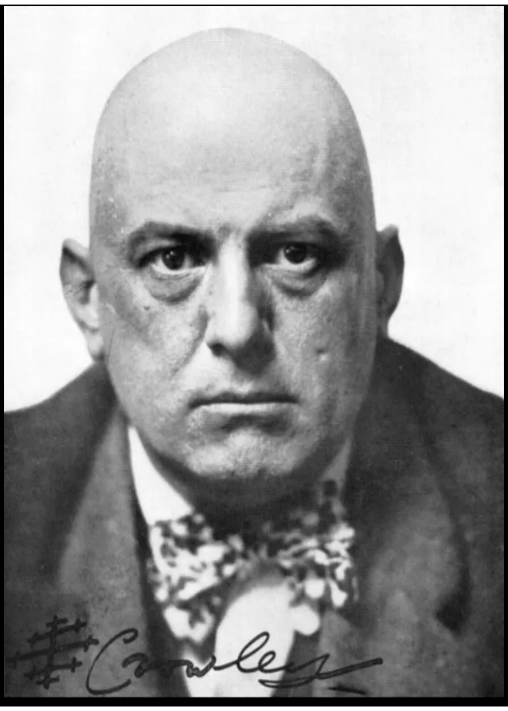
LEARNER GOALS:
- understand what an algorithm is vs LLM
- understand what users need to work with and understand them
- know what scares people about them and machine learning
- what to do when they go wrong
- how to think about liability

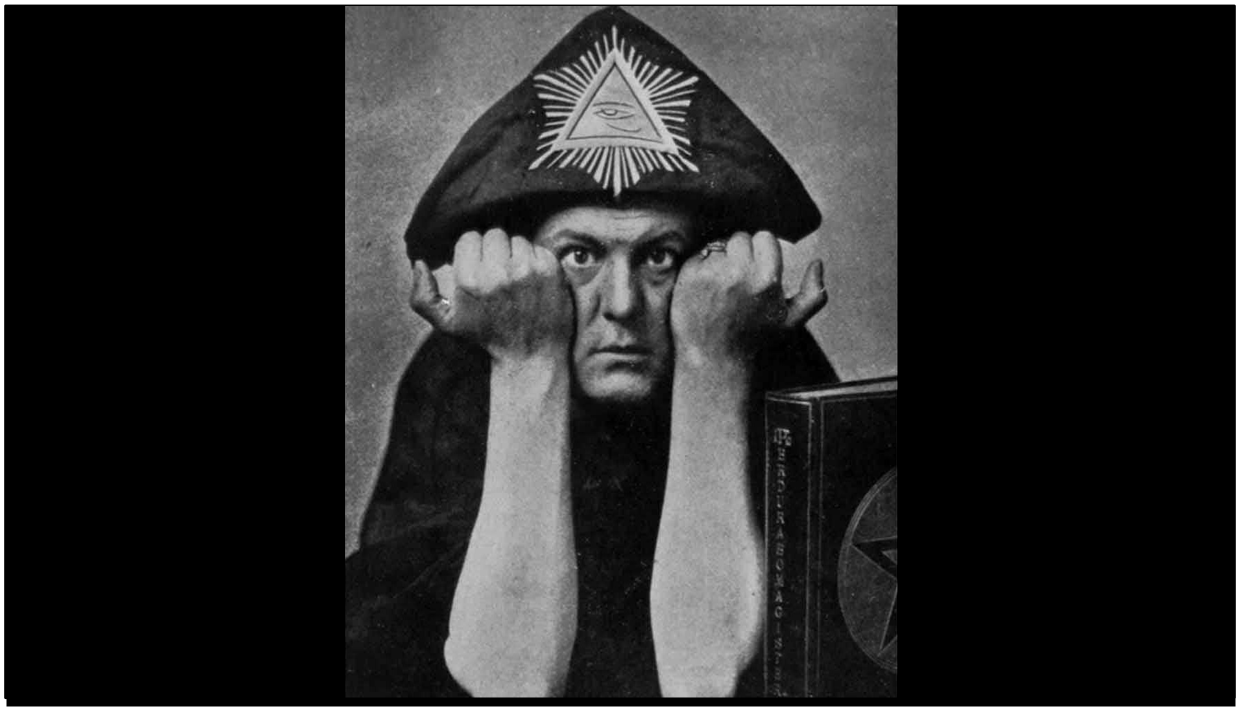
hi everyone, good afternoon. what a couple of days, huh? thanks for sticking around long enough to listen to me.

let's talk trust in algorithms, and what we can do to keep people comfortable.

i'm calling this one — _it came from the black box!_



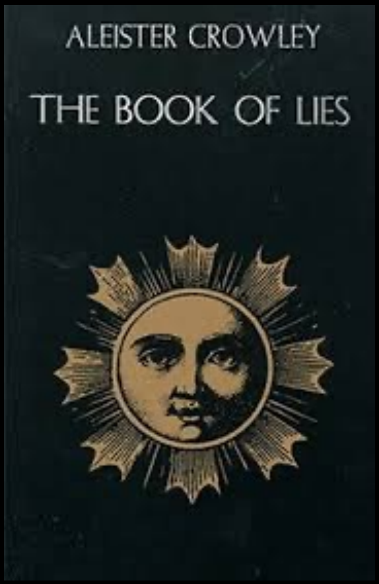
if there's one thing i want to leave you with today, it's this: don't be alastair crowley.



aleister crowely was born in 1875 and pretty quickly decided that everything about his life had to changed.

he rebelled against his parents, the institutions around him, and especially his religion.

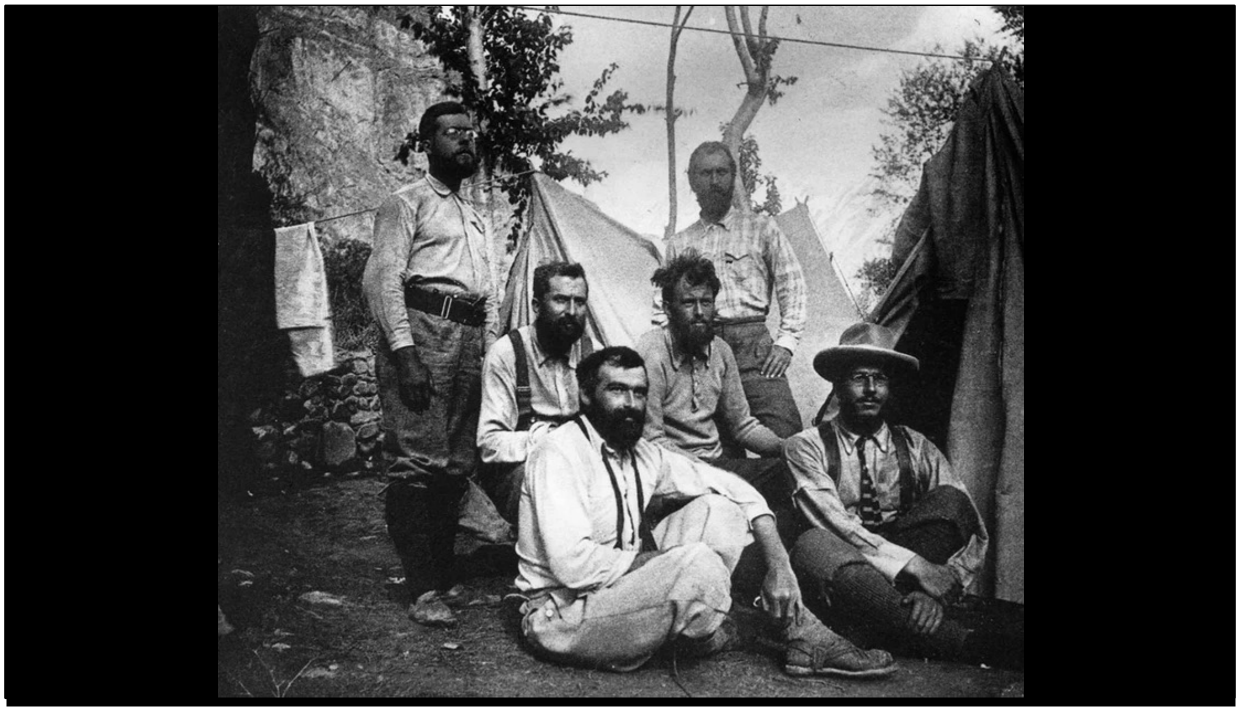
his mother called him "a beast," a moniker he adopted and became the self-styled "Great Beast 666."



eveyone else called him "the wickedest man in the world.”

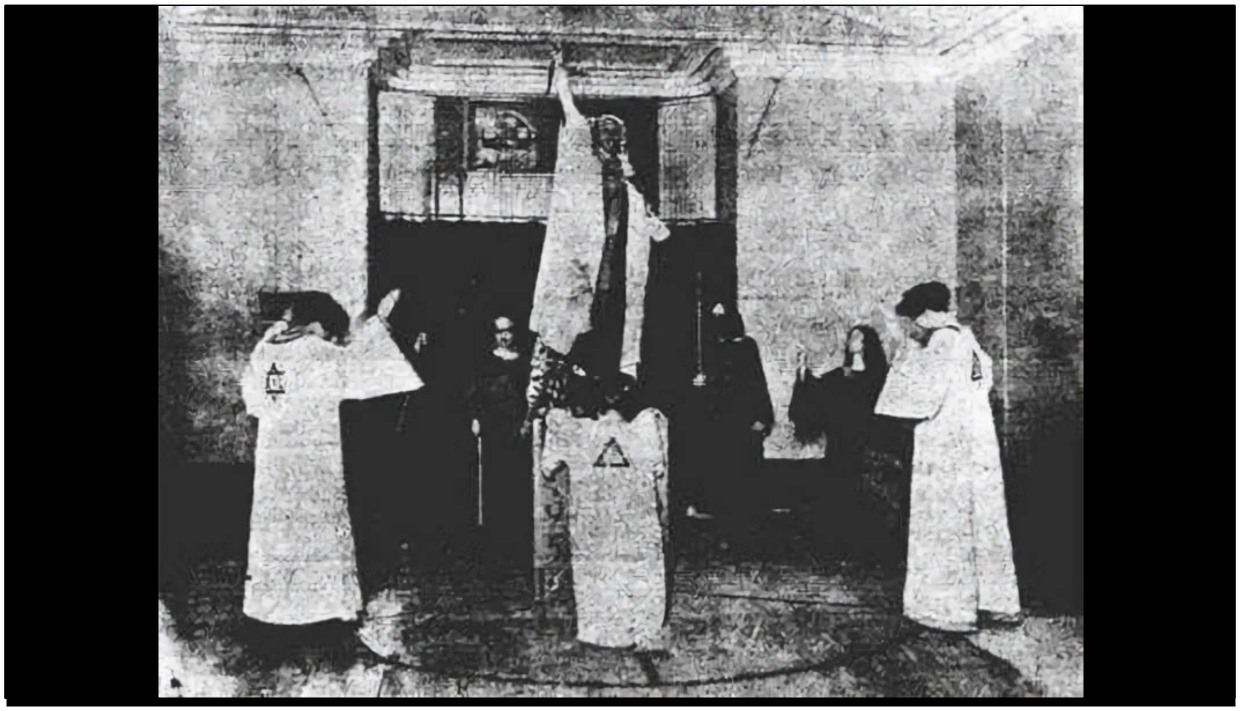
he created the religion of thelema that had one law: "DO WHAT THOU WILT."

he bent people to his will, breaking apart marriages so he could sleep with whatever woman he wanted — whether they wanted it or not.



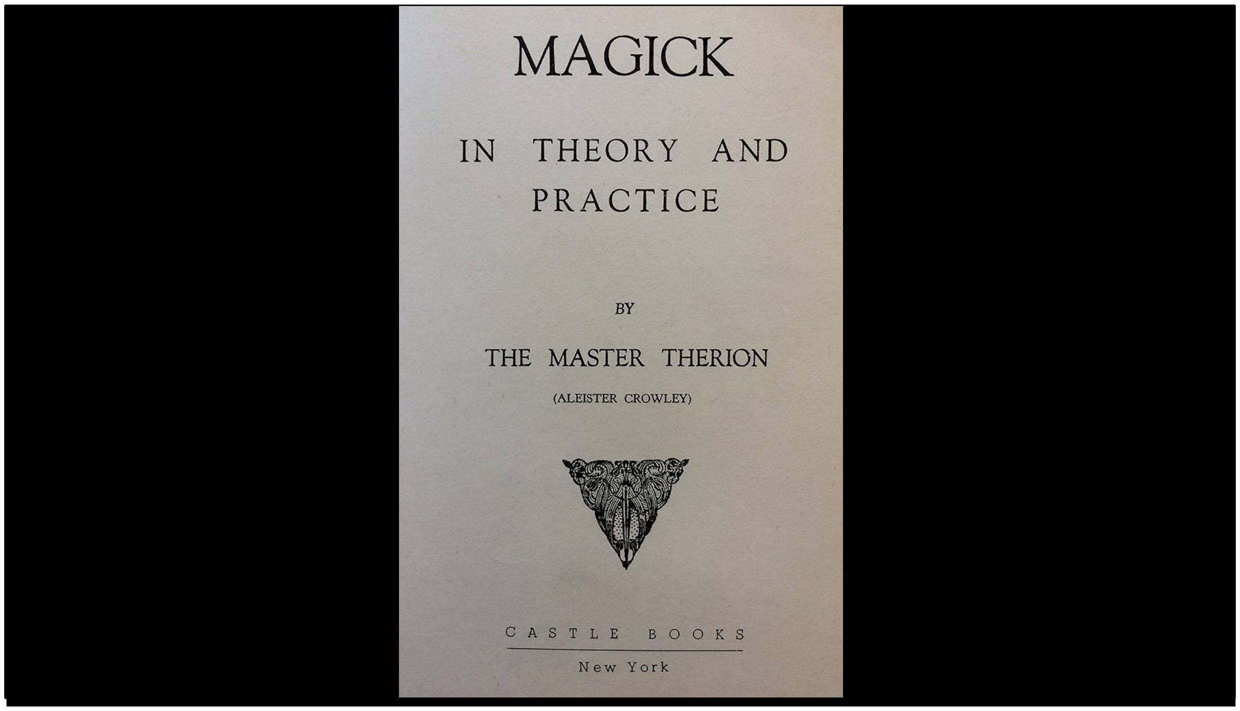
when mountaineering, he — that’s him second from left — and his climbing group were trapped in a storm.

crowley left them to die — the rumor is that he was drinking tea while they begged him for help that he refused to provide.



one of his followers drank cat's blood. her husband later died when he drank from polluted water — either because he was forced to or because it was all he had access to.

many of the stories about crowley were probably overblown — it's unlikely he sacrificed children —



MAGICK

IN THEORY AND PRACTICE

BY

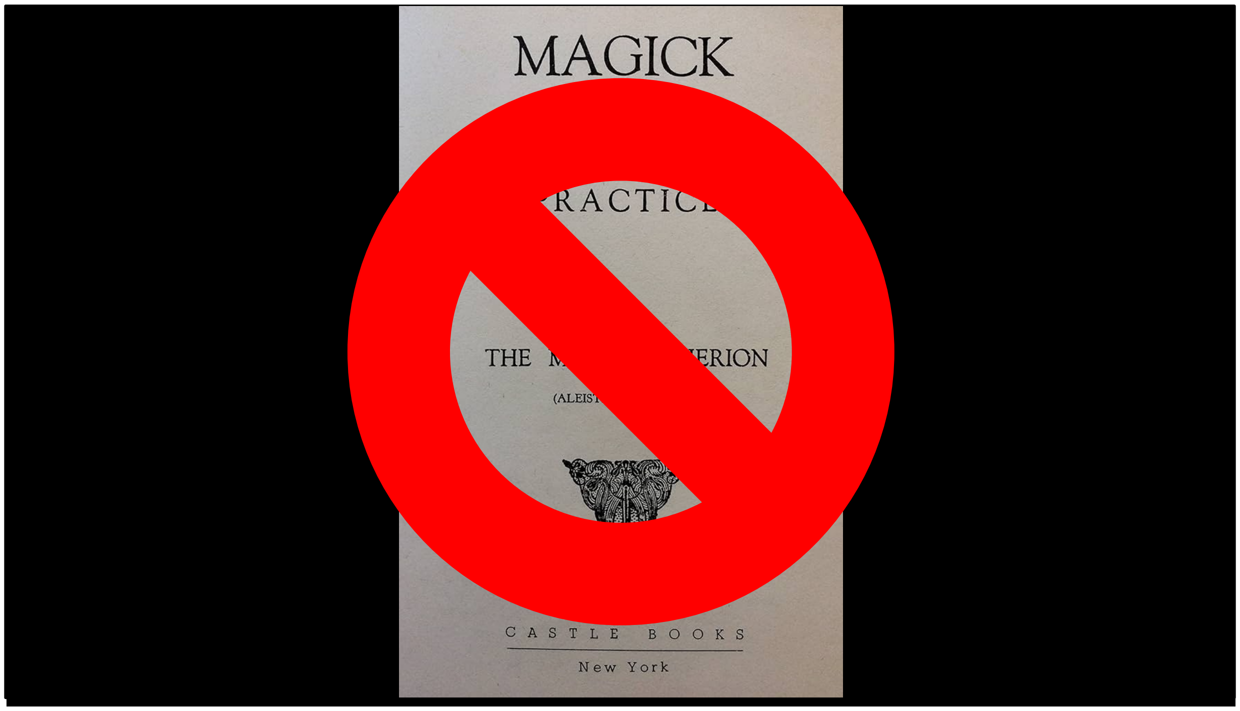
THE MASTER THERION

(ALEISTER CROWLEY)

CASTLE BOOKS

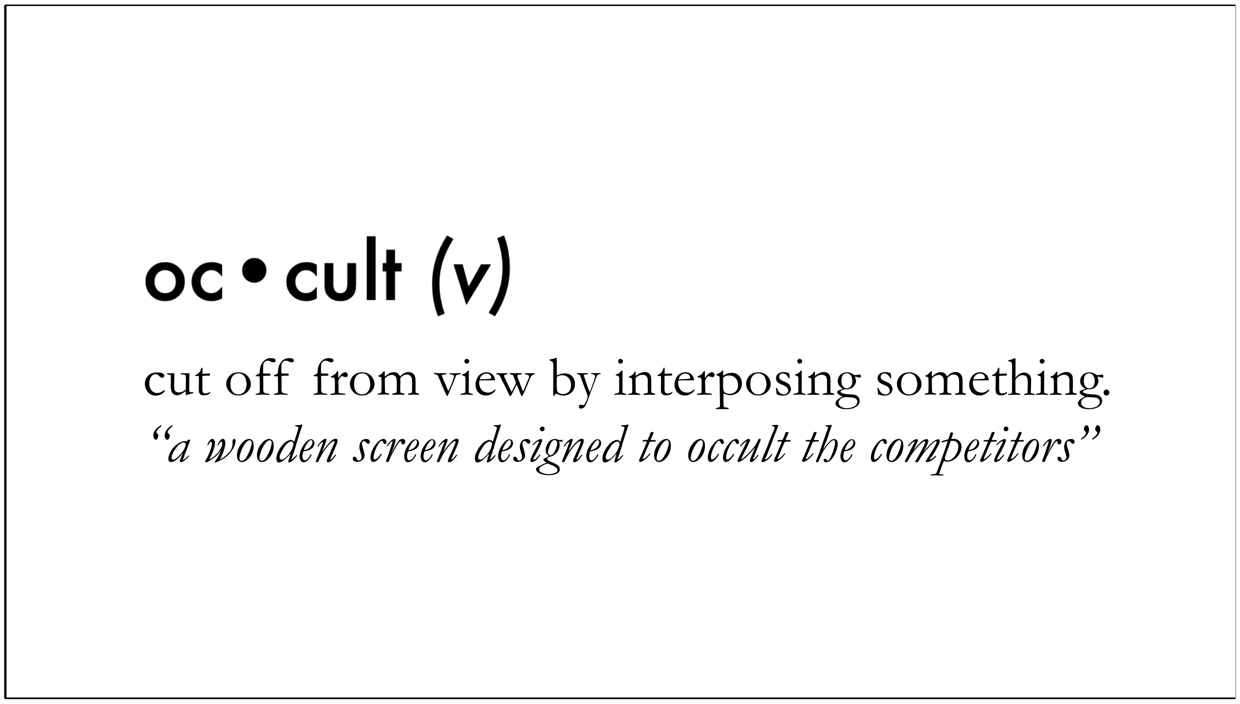
New York

but the crux of it is crowley's embrace and usage of the occult were what gave him his power.



not in like, a magic way, though.

what's so insidious about "the occult" can be found in the meaning of the word itself



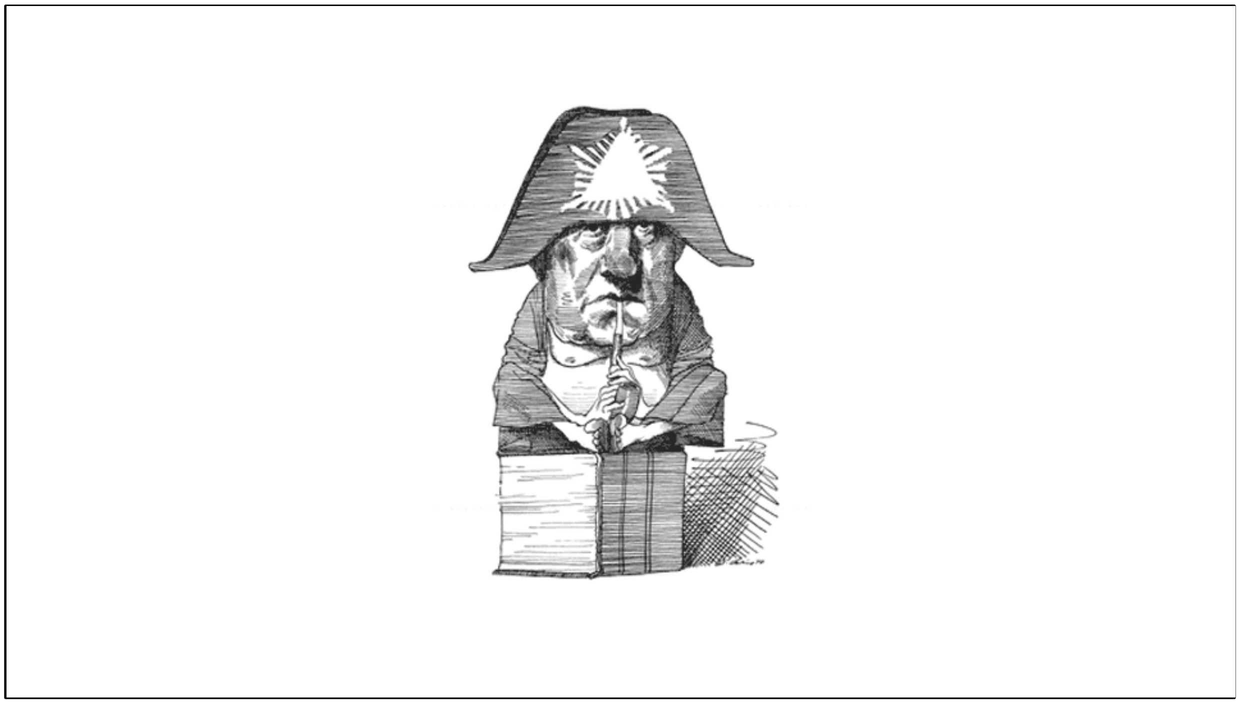
oc•cult (*v*)

cut off from view by interposing something.

"a wooden screen designed to occult the competitors"

— it's _hidden_.

if you have an occult diagnosis at the doctor's office, it's something that doesn't have a lot of signs or symptoms.



crawley was so dangerous because he was the only one who could interpret the meaning of the hidden supernatural — his followers had to trust him completely with no proof.

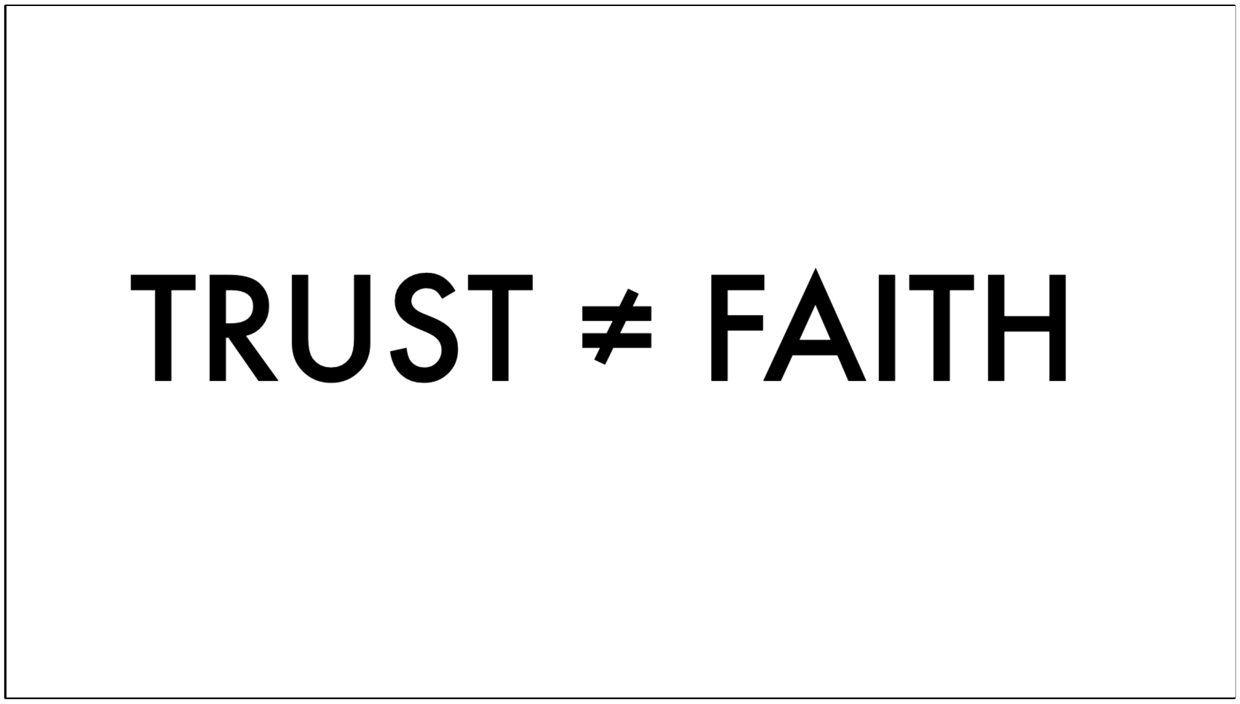
and that's dangerous! it leads to systems that are rife for abuse.

it leads to situations where people are making decisions based on nothing more than what they believe about a system, not what's _true_ about it.

it leads to situations where you drink cat's blood, because this strange, demon-obsessed man told you to.

if your users ever get into a situation where they say "i don't know how it works, but i keep using it" —

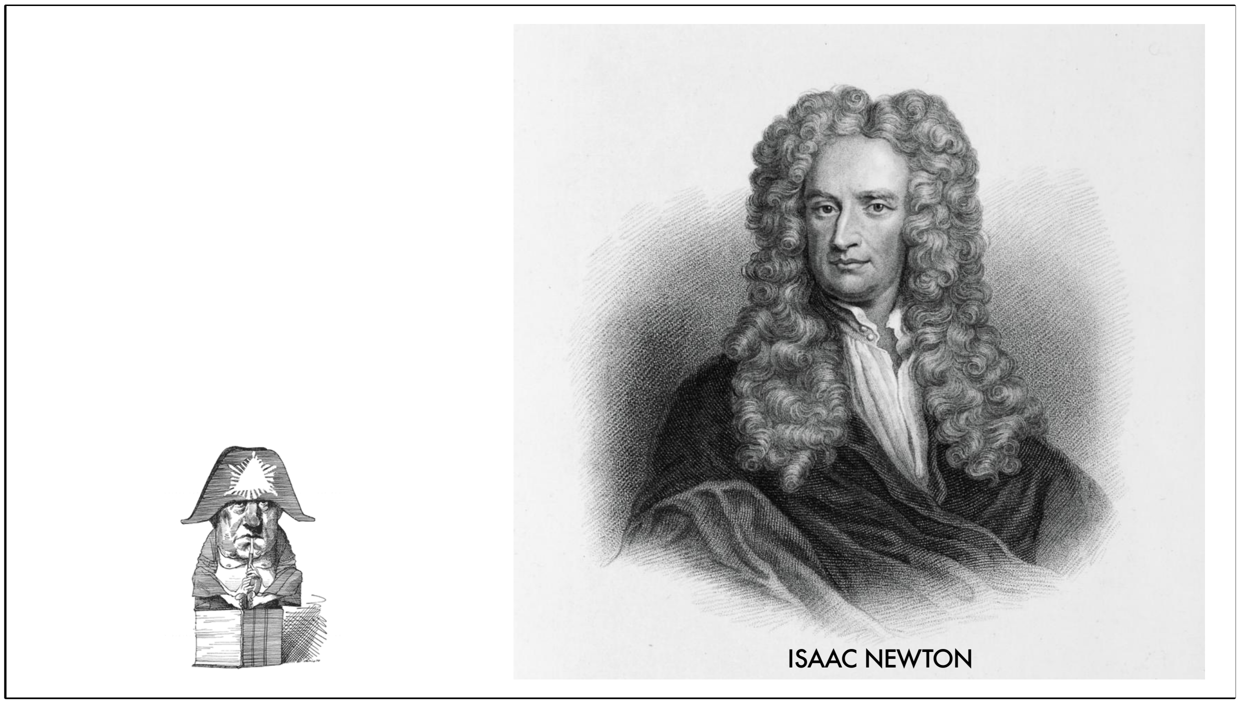
Congratulations, you have entered the realm of the occult. prepare for a sacrifice, either in the form of a patient or a lawsuit.



TRUST ≠ FAITH

trust without knowledge is faith.

and while that's a fine thing, we don't build medical devices to operate on faith.



the opposite of the occult — what reveals hidden things — is information and understanding.

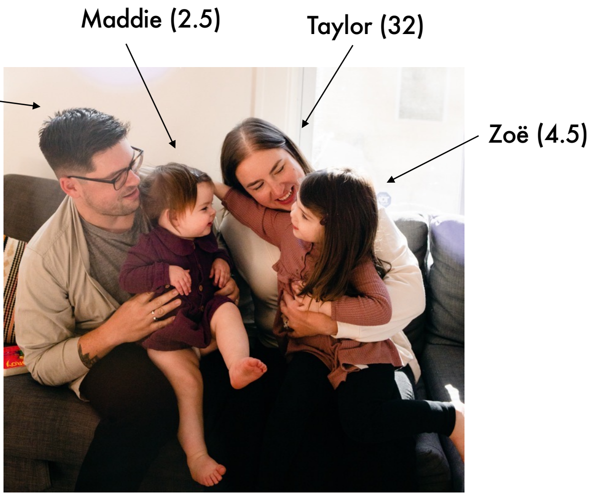
our job as designers is to provide our users the knowledge they need to understand how an algorithm works.

even if they're not working with it, it is _impacting_ the patient and thus their responsibility.

i'm really passionate about this, and let me tell you a little about who i am and why i'm here today.

ANDREW LILJA

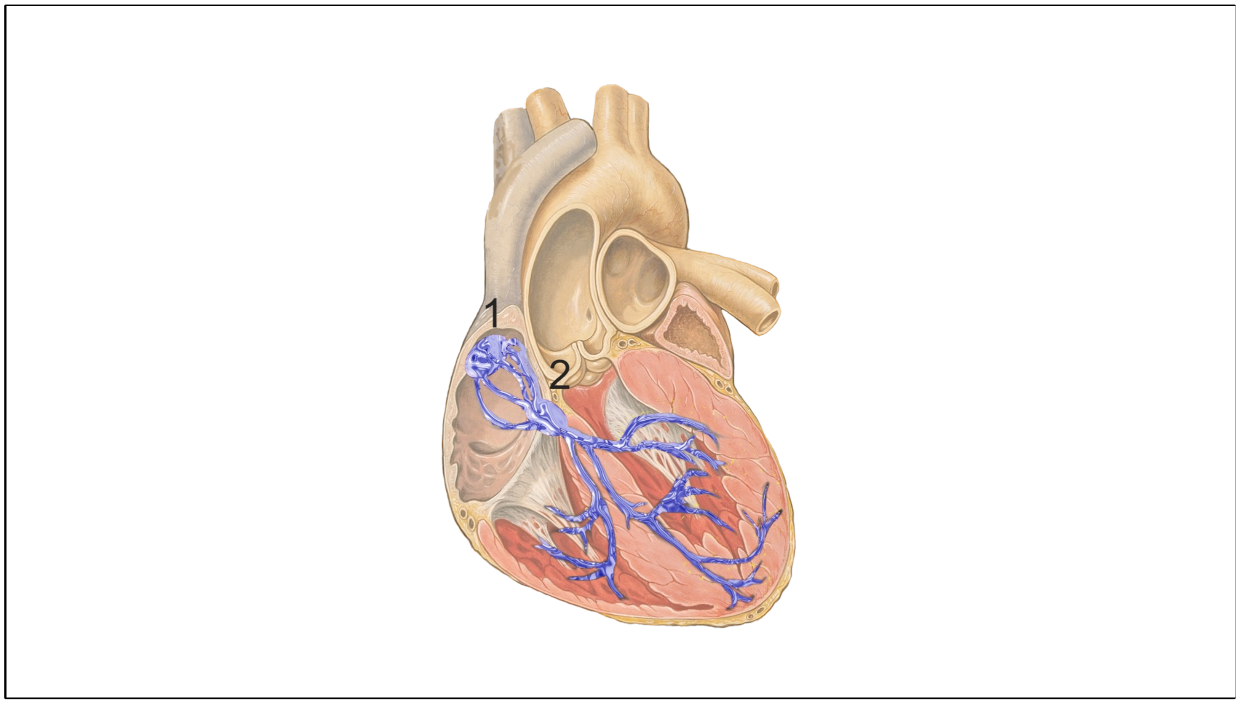
- BA in psychology
- MS in human-computer interaction
- Principal Human Factors Design Engineer
- Very passionate about things that are boring to talk about at parties



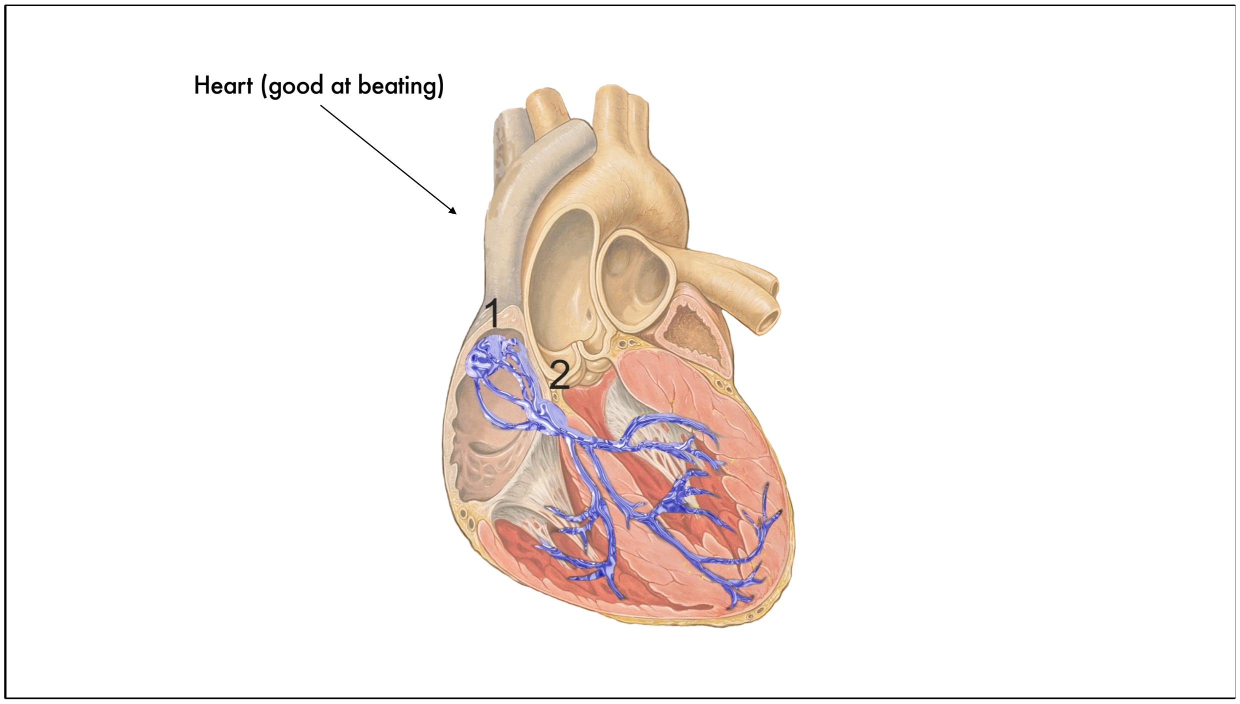
so — hellooooo. my name is andrew lilja, and i'm a principal human factors design engineer at medtronic.

i work in the cardiac rhythm group, which means it's my job to build things that keep people alive when their hearts don't want to anymore.

we do this in a whole bunch of ways, but i'm mostly responsible for the ones involving putting a bunch of electricty into them.

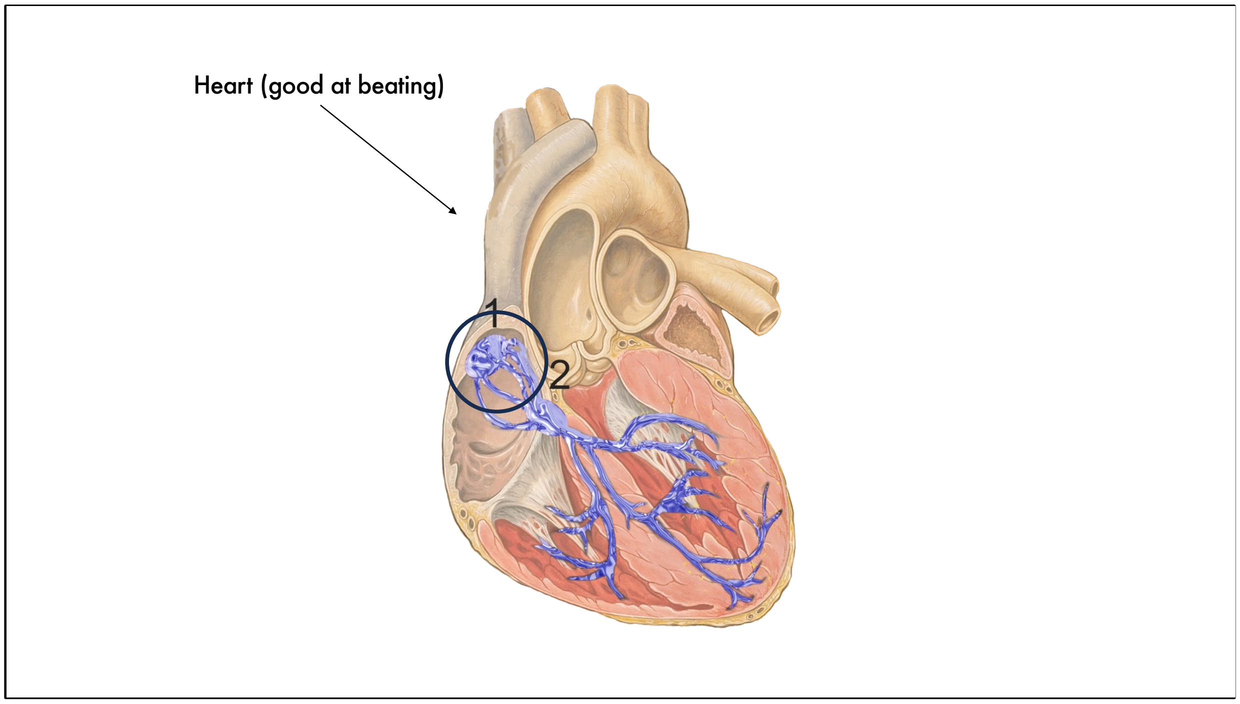


this might come as a surprise, but hearts are actually very good at beating.



this might come as a surprise, but hearts are actually very good at beating.

you have a little part of it right here



called the atrioventricular node that helps organize electrical signals from the brain and turn them into heartbeats.

and it's so good at this that you can actually remove a heart from the body _completely_ and it will still beat on its own at a steady 60 beats per second.

but sometimes things go wrong.

there might be physiological issues with the nerves or the muscles,

there might be something wrong with the connection between the brain and the heart —

but most of the time, the outcome is that the heart doesn't beat in the right way. and since we are electrical creatures,

you can use a little jolt of electricity to make the heart beat again. do it at a steady rate, and people can stand up and walk and be alive again.

and this works really great for about five minutes.

then the patient needs to run, or their heart starts beating on its own again, or their heart decides to really throw us for a loop and go into a fatal arrhythmia.

these are things our basic pacemaker needs to expect and interpret, and we use a lot of algorithms to do that.

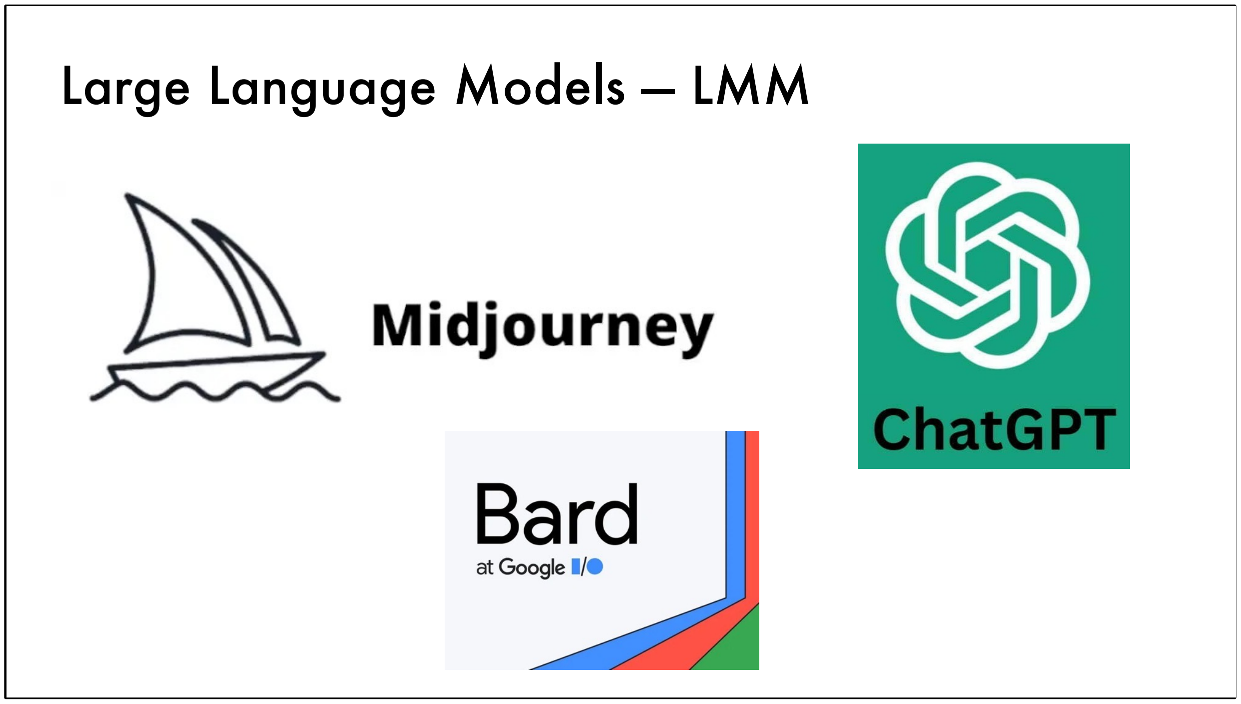
but — let's take a second to get our terminology straight.

there is a lot of talk these days about "artificial intelligence."

this so-called AI definitely relates to algorithms and what we work with, but let's get clear about what all these things are.

personally, i hate the term AI.

it's so generic and general that it's practically useless to discuss with.



Large Language Models – LMM

Midjourney

ChatGPT

Bard
at Google I/O

it's so generic and general that it's practically useless to discuss with.

what most people are thinking about when they say "AI" is what's called a "large language model,"

which at their very core are statistical models that have been given enough data so they can make good guesses about what words or pixels belong next to each other.

there are many methods of doing this, but the common demoninator between all of them is that they are creating _statistical models_ and then using that to take novel input and produce things that are similar.

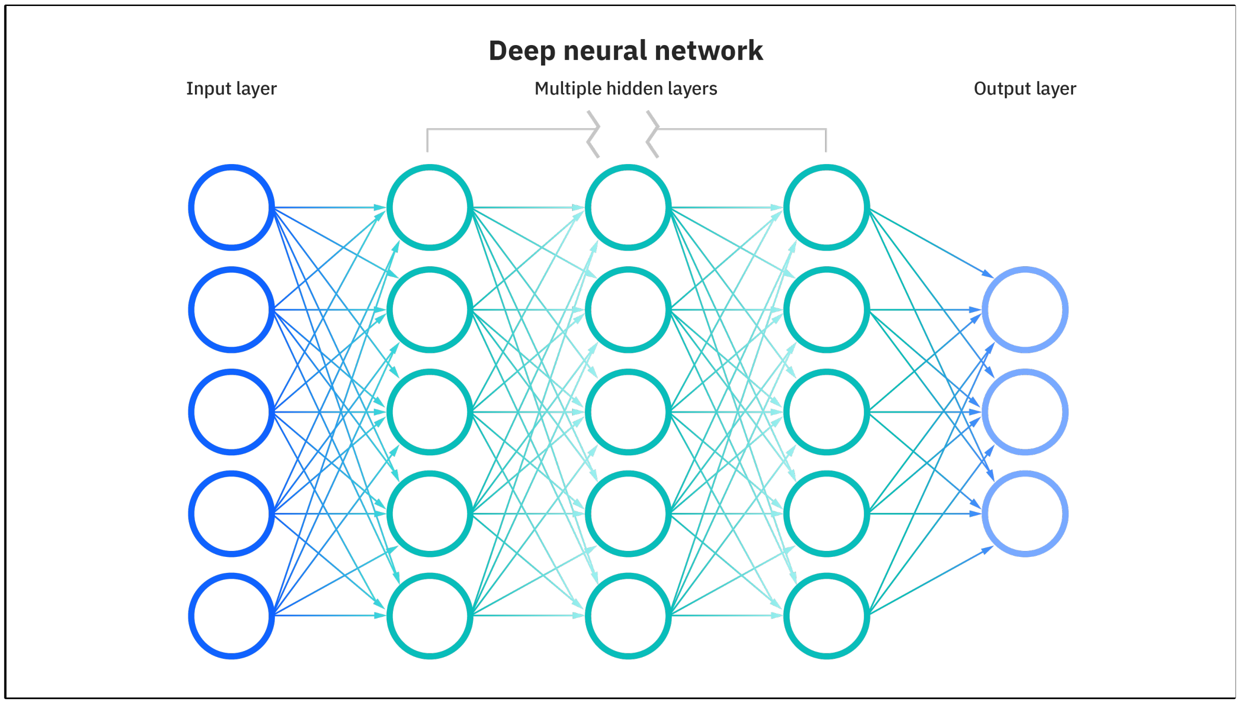
Midjourney

“Draw me an adorable cat.”

you feed it a bunch of images that have been tagged with things like "cat" and "box" and "very adorable"

and then when you give it the words "draw me an adorable cat" it can spit one out for you.

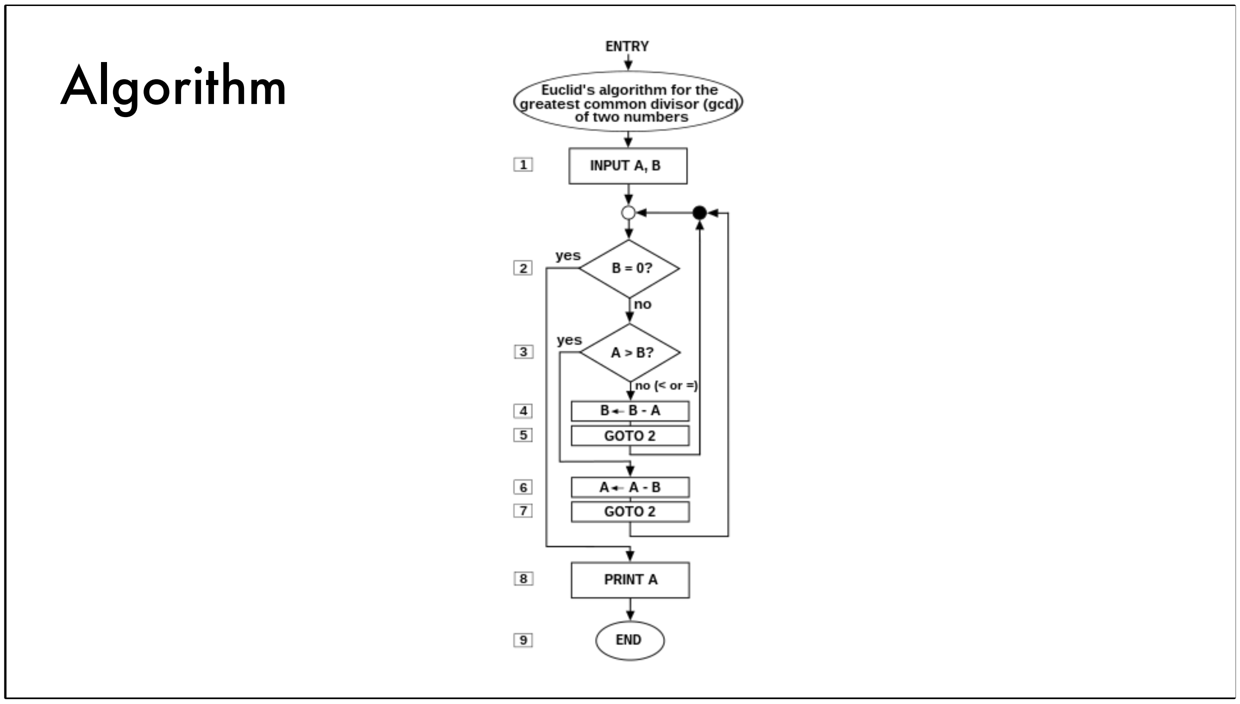
the more pictures you gave it during "training" and the better your descriptions of them were, the more options you have down the line.



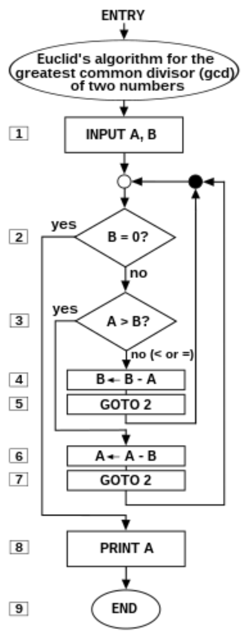
so those are LLMs. they're a subgroup of a field called "machine learning," which is exactly what it sounds like.

but all of them are basically black boxes. you understand the model — how they were built — but you have no insight into _why_ it's making the decisions it did.

they're just a huge network of statistical weights — there aren't any rules that you can look at and understand.



Algorithm

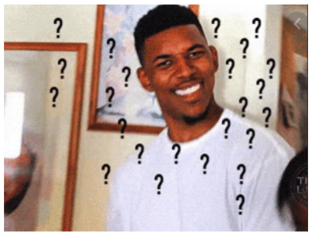


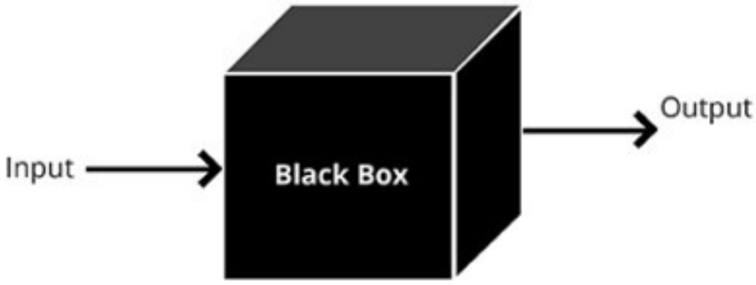
compare this to an _algorithm_, which is a set of explicit, described rules that produce _predictable_ outcomes.

that's the major difference: you can predict the precise outcome of an algorithm because you wrote the rules to make it work.

given the same input, you can predict what the output will be.

if you get an incorrect or weird output, you can go to the algorithm and figure out what happened and fix it.

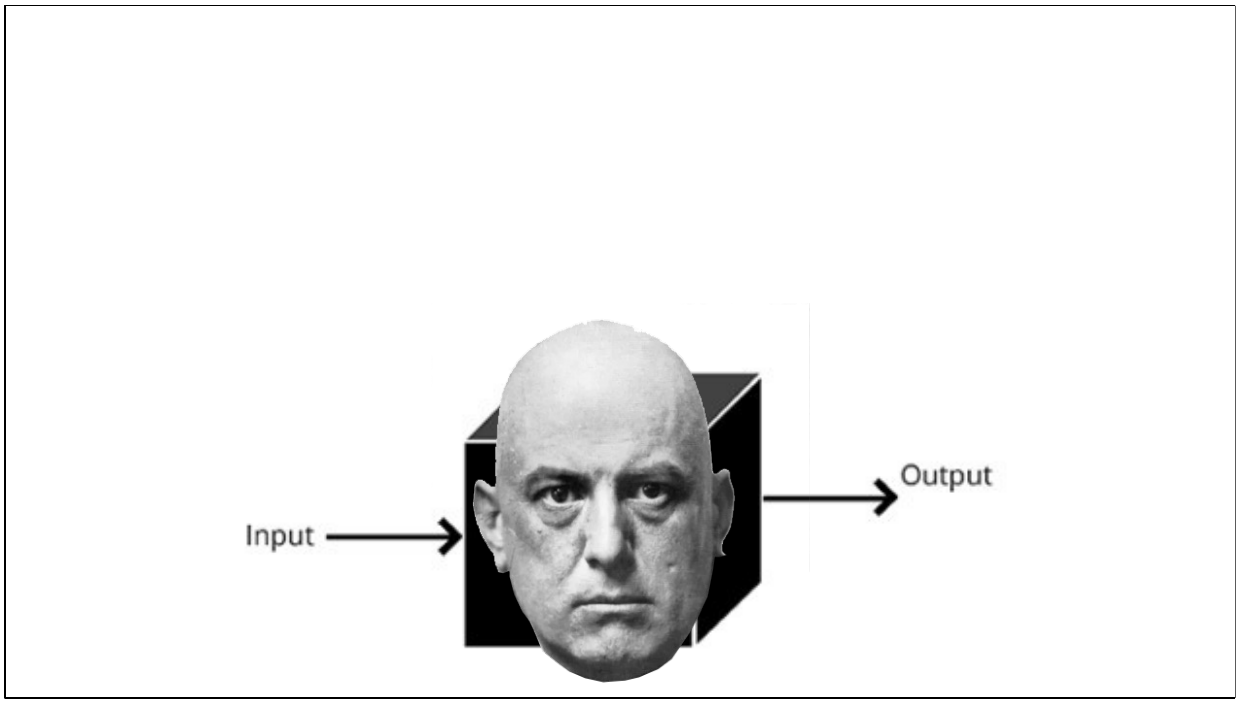




an LLM — and ML generally — can create surprises.

if it does something you didn't expect, you can't figure out why. all you know is that the group of inputs you gave it turned into that outputs.

if it came out a wrong or unexpected way, you can't know why. maybe your training data was polluted. maybe your input was bad. you don't know! you can make some guesses, but you don't know why.



it's an occult technology.

you just have to trust that what goes in is going to give the correct output.

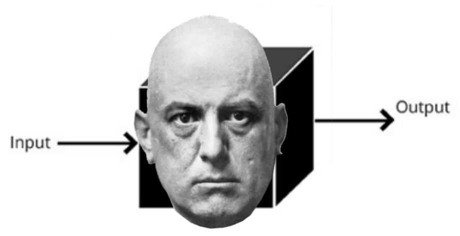
and sometimes that's what you want! if you want something that's novel and entertaining, an LLM may be just what you're looking for.

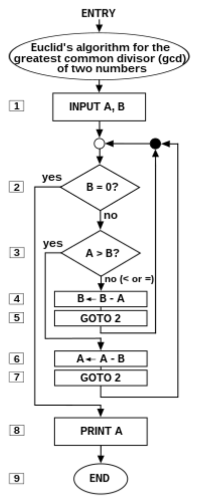
but if you're using this occult AI black box to determine when a patient needs a pacemaker beat, or to develop care plans, or do anything at all that could result in harm — don't do it.

use an algorithm instead.

KNOWABLE

UNKNOWABLE





The crux is that one is unknowable, and the other is not. And if your users *know* how it works, they can trust how it works.

You want your users to know. You want them to be able to make predictions about outcomes.

When they understand how it works, they can use it in their work. When it stops being mysterious and confusing, it starts to become usable.

Things that are unknown are scary. You don’t know what’s happening, why the algorithm is doing something, how it can be controlled or managed or used.

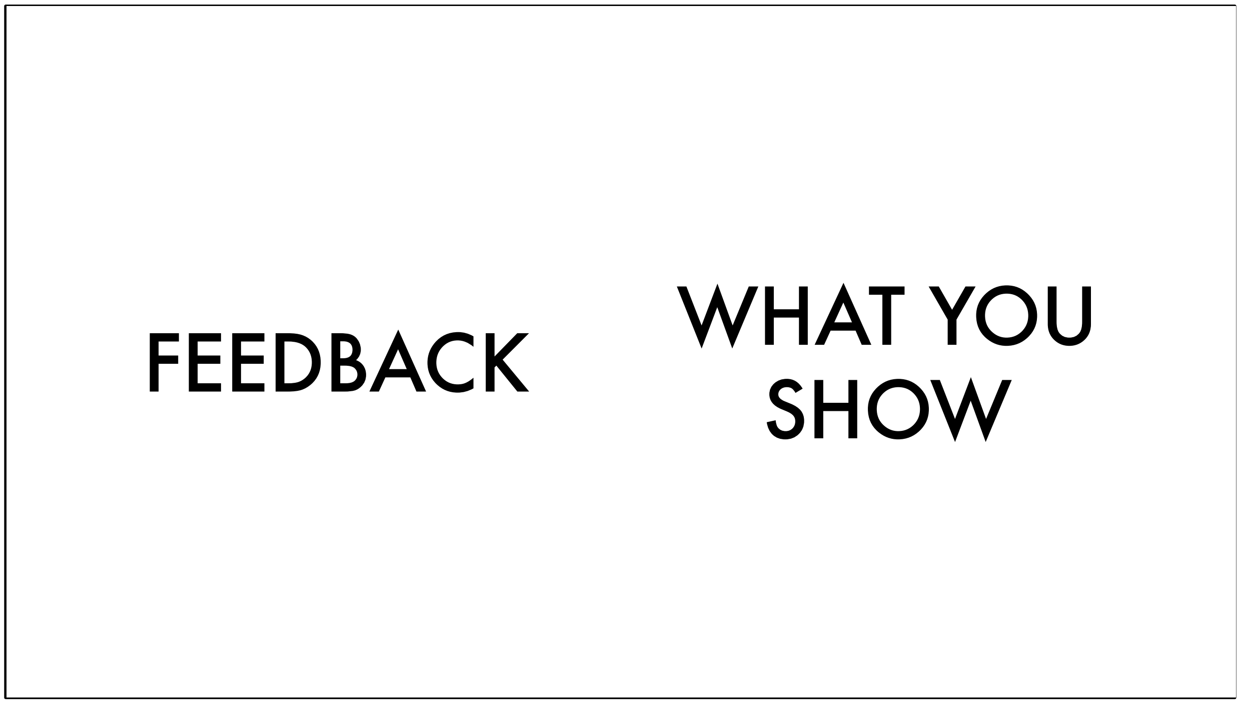
becomes understood and you can figure out how to leverage it. It might even become as boring as an office hallway.

It stops being occult and starts being understandable.

so if the opposite of the occult is the known, the visible —

how do you do that? how do you build trust in something?

i think there are two main ways we do it:

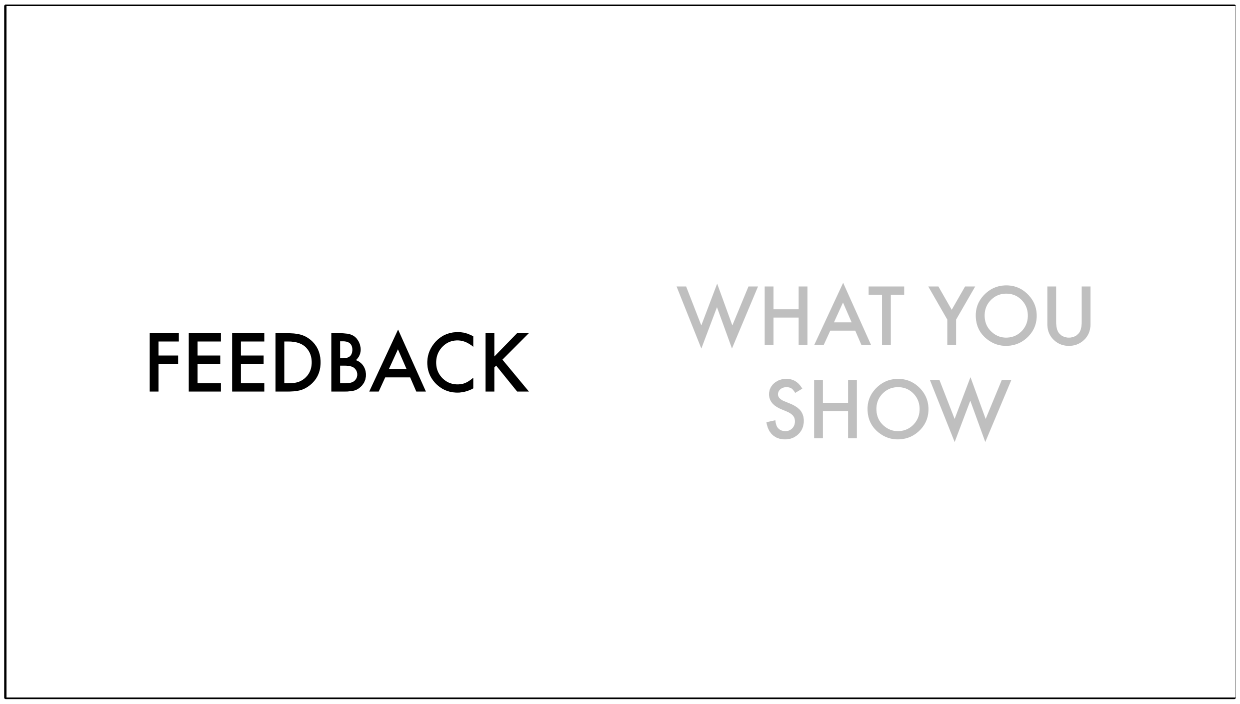


FEEDBACK

WHAT YOU SHOW

via feedback and revealing the right complexities.

and your job is to use those to make sure your users understand how the algorithm works, and how they can use it in their workflows.

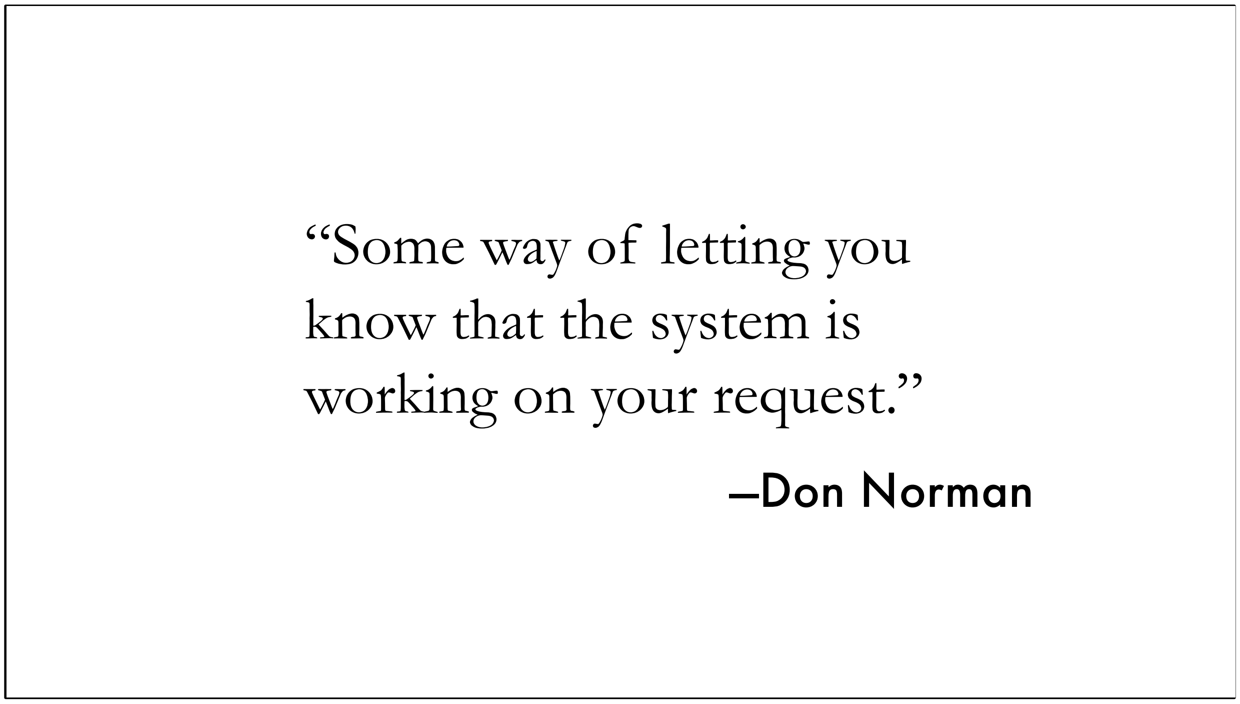


FEEDBACK

WHAT YOU SHOW

let's talk feedback first, because i think it's simpler.

feedback is so fundamental to interfaces that don norman literally defined it.

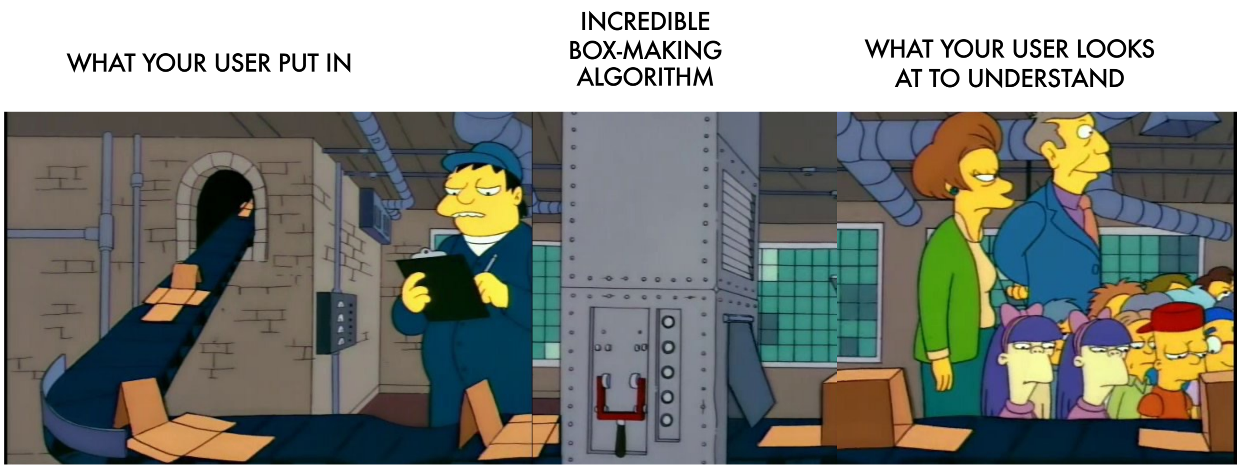


“Some way of letting you know that the system is working on your request.”

—Don Norman

i want to take this one step further and add that it's not just letting you know that the system _is_ working on your request,

it's also telling you _what the system is doing_ with your request. it's saving, it's searching, _whatever._


WHAT YOUR USER PUT IN
INCREDIBLE BOX-MAKING ALGORITHM
WHAT YOUR USER LOOKS AT TO UNDERSTAND

when it comes to an algorithm, this is about predictability.

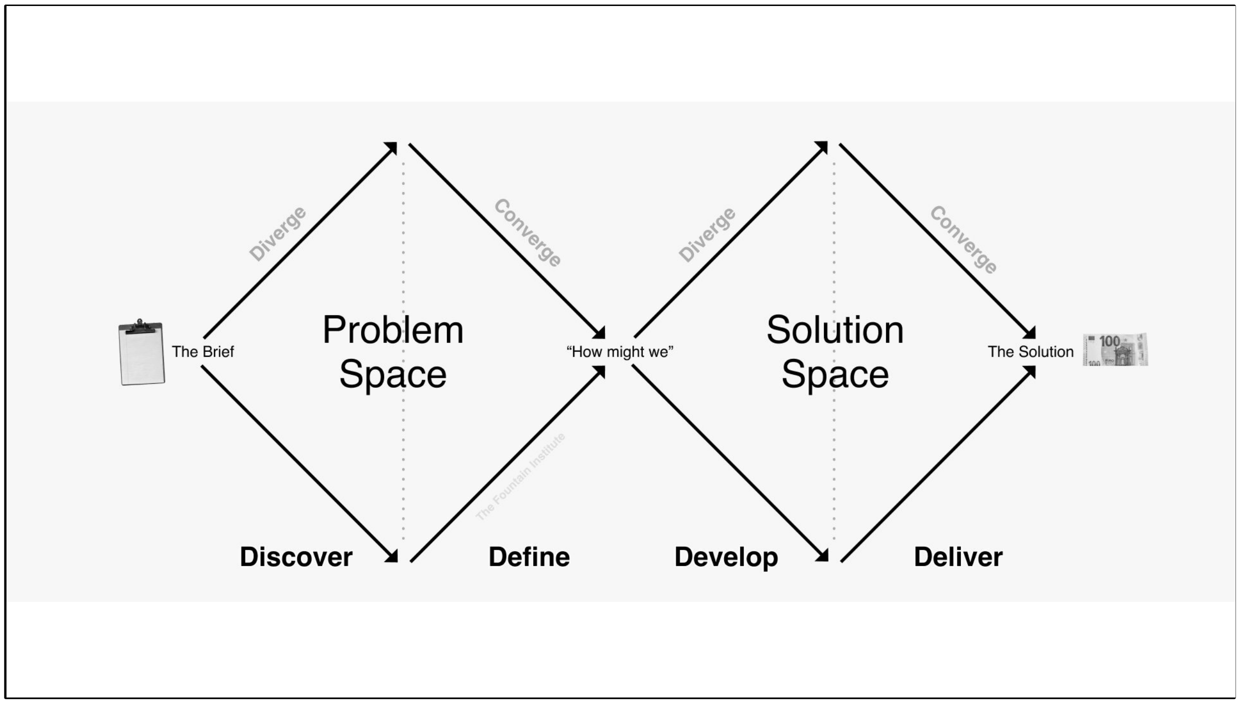
you put something into the algorithm and it comes out giving you an answer you expected — or at least within the range of spaces that you expected it to come from.

You need to show your user what's going on, how those inputs are being interpreted.

Your user sees flat paper going in, something happens, and then a box comes out — every single time. They use that feedback to build a mental model of what's going on inside the algorithm.

It lets them leverage the algorithm more effectively, too. When they see what comes out, they can learn how to put better things into it so they can get better outputs.

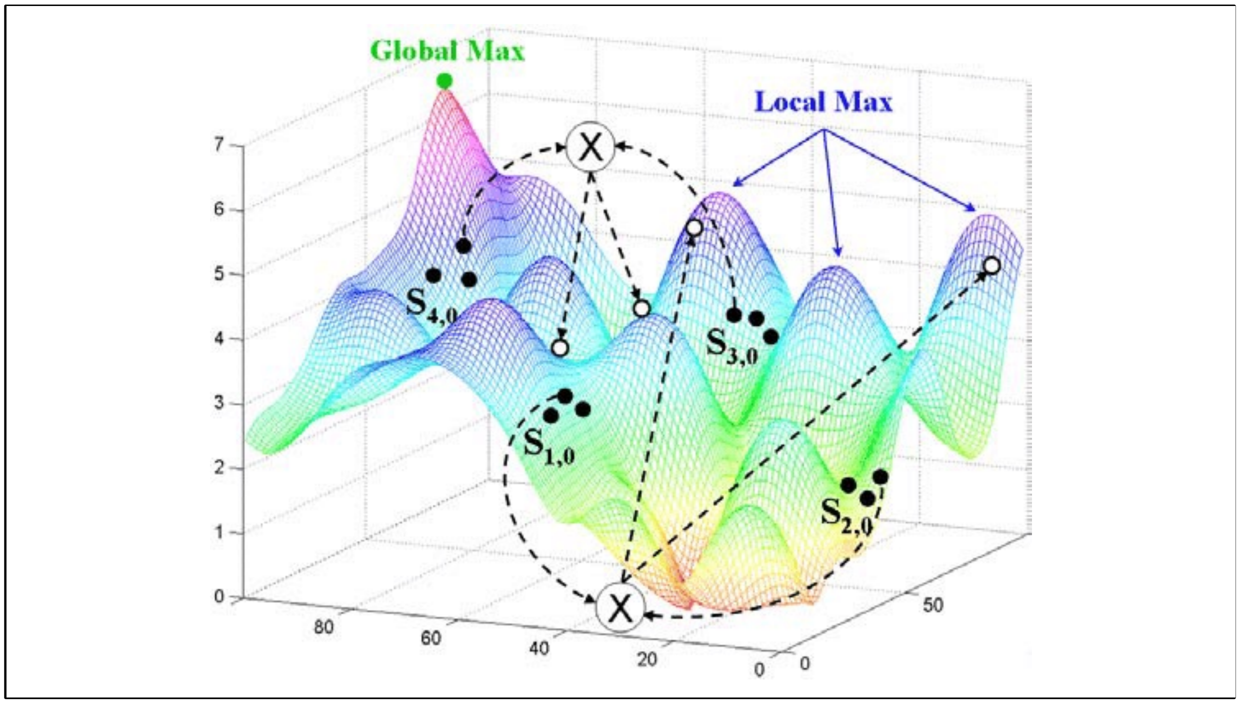
You need to do this in your interfaces! You need to reveal what the output is.



you know, one of the things we do as designers and researchers is define a "problem space" —

given all the constraints and stakeholders and inputs to a problem, here's the things the problem could be.

the problem we'll solve comes from this space.



i'd like you think of the reverse: and algorithmic "answer space" —

given the inputs and all the rules of the algorithm,

you know what a likely outcome from the algorithm is going to be. your users will be developing this all the time.

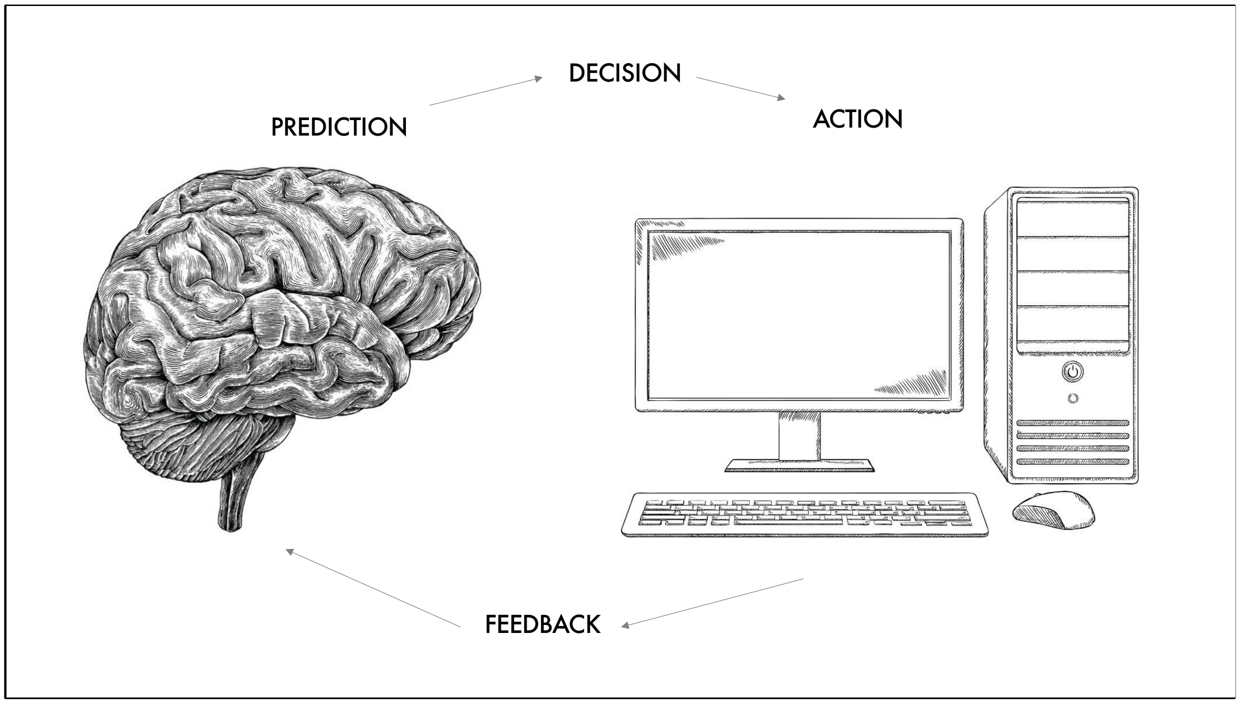
every time they get an output back out, it will further define this answer space.

i'll spare you from the details right now, but what's happening is that your users are developing mental models of how the algorithm works.

and you simply cannot stop them from doing that. our brains will do it with just about anything.

we are pattern-finding machines, which is why you look at this picture and see a face and not a pile of vegetables.

Your users will grab onto any scrap of feedback they get from the algorithm and use it to make decisions about whether or not they can rely on it.



mental models are _sticky_ — once a user has a general idea of how the algorithm works, it's really hard to convince them otherwise.

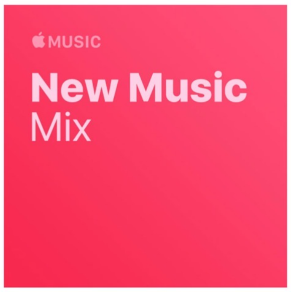
so the only way to make sure they're getting it correct is by making sure the feedback — the answers — they're getting are correct.

so an algorithm's outputs will exist in this answer space.

your users begin to develop a model of what are likely outputs given certain inputs.

and your users will trust it more when it's predictable, when they develop an understanding of how input A will turn into output Z.

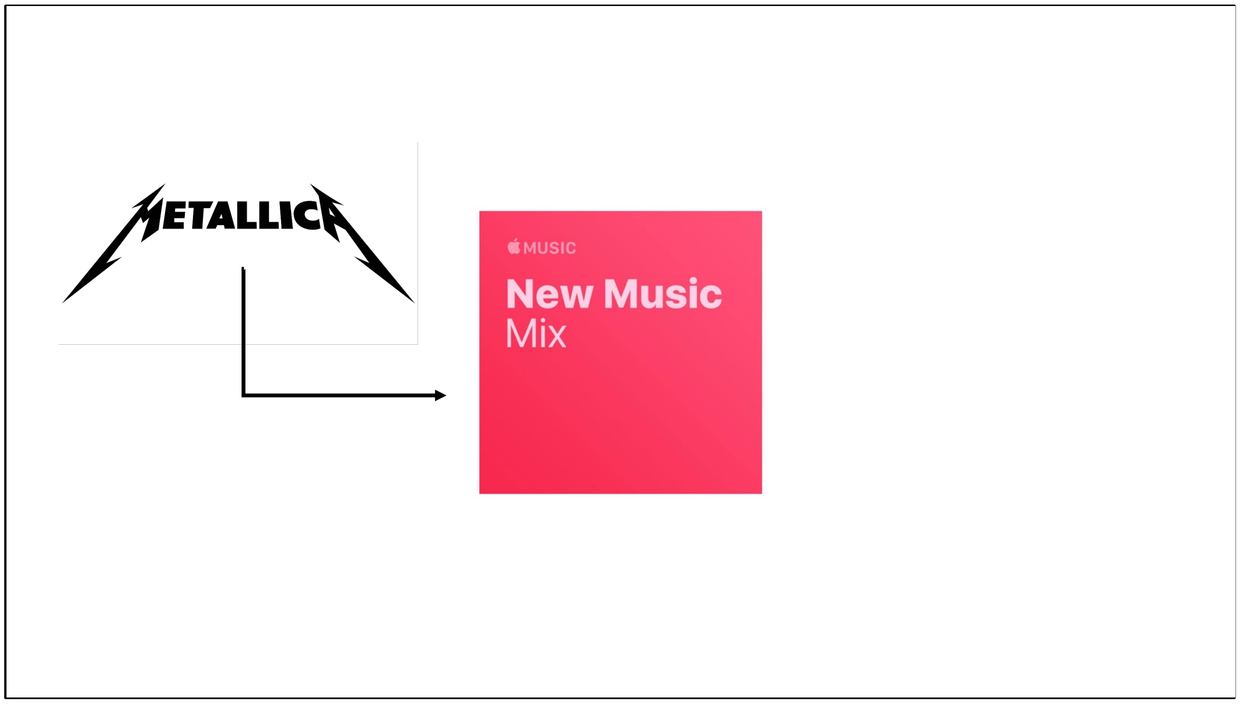
A really familiar example is music recommendations



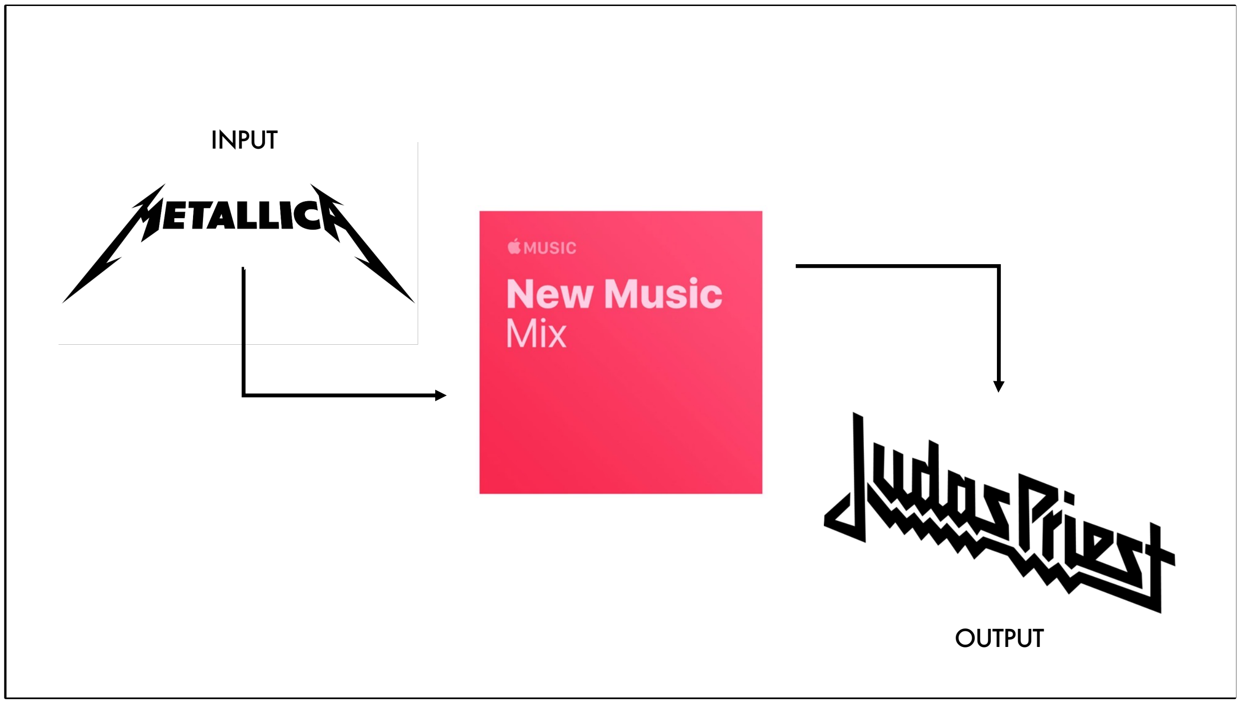
in spotfiy, apple music, whatever, all these platforms have automated "discovery" features.

and we have a really intuitive sense of how they work — you listen to a lot of music of one genre or artist, and it will recommend you similar things.

the algorithm is providing us feedback via its output about how the things we put in (what we listen to)

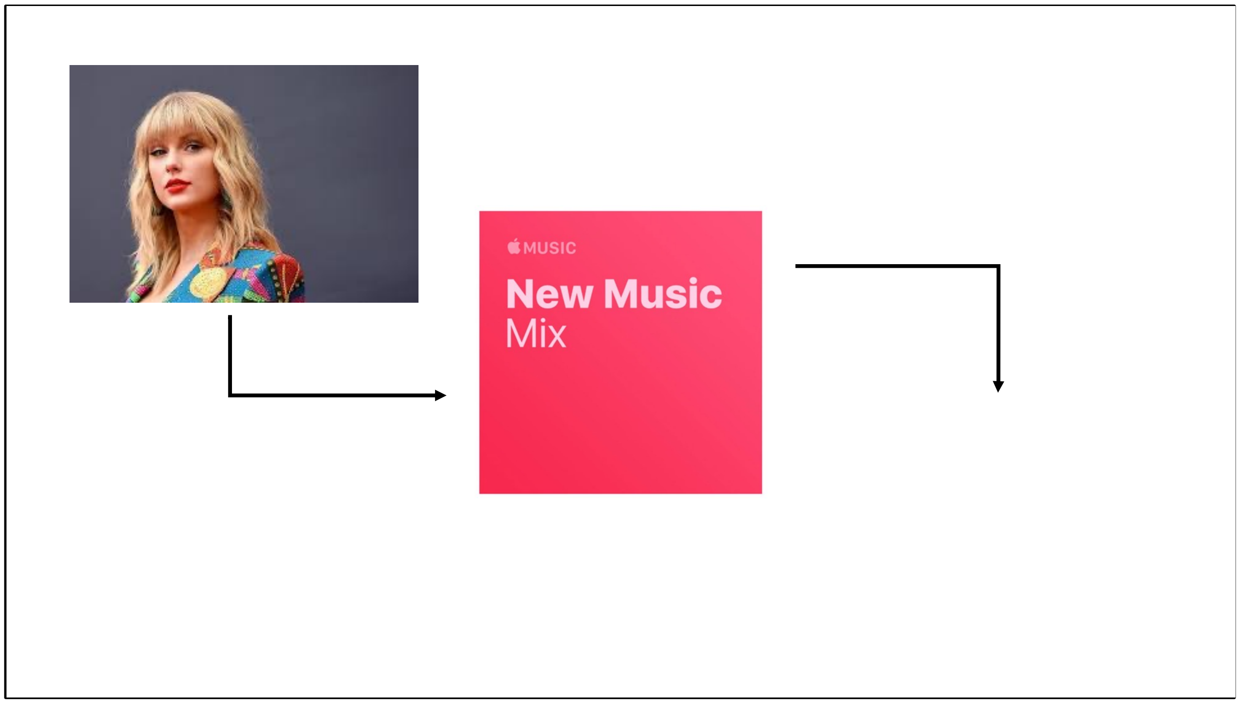


generate the things it gives us (what it recommends)



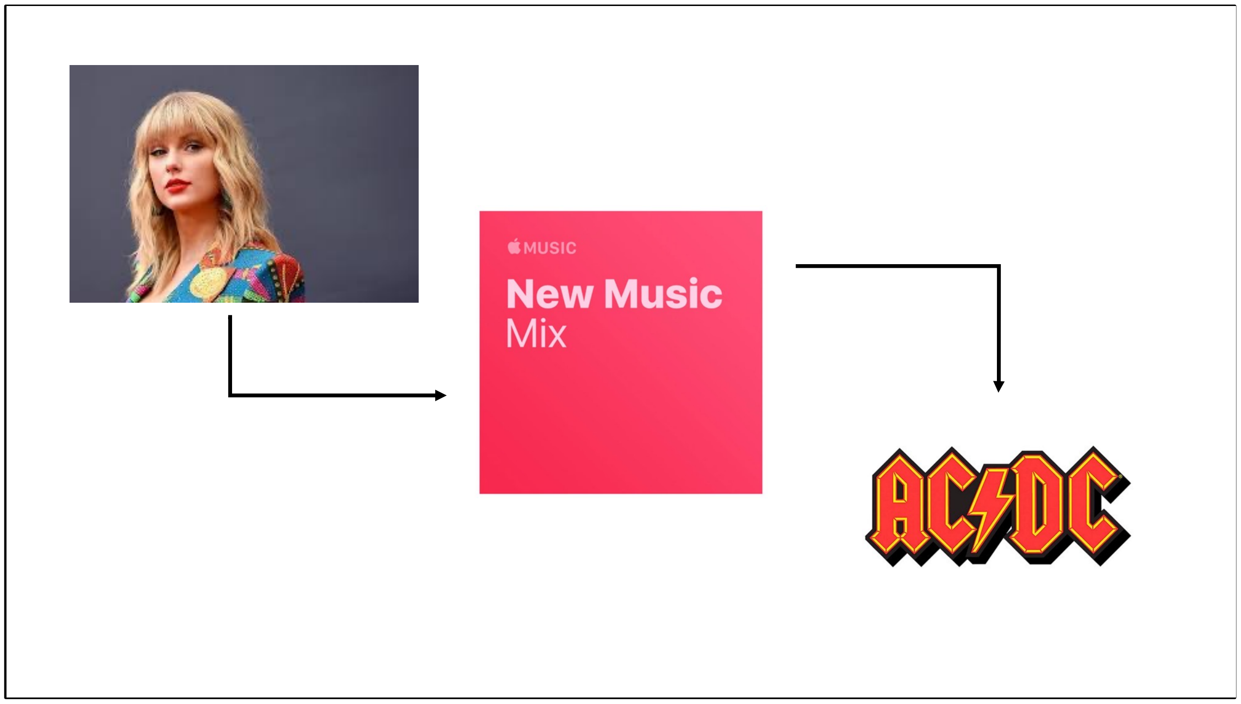
This is wonderful and very predictable. People like this algorithm because it gives them interesting things and makes their lives easier. And it's predictable!

Ask anybody on the street how this works, and they can at least tell you how their inputs — the music they listen to — gets transformed into outputs.

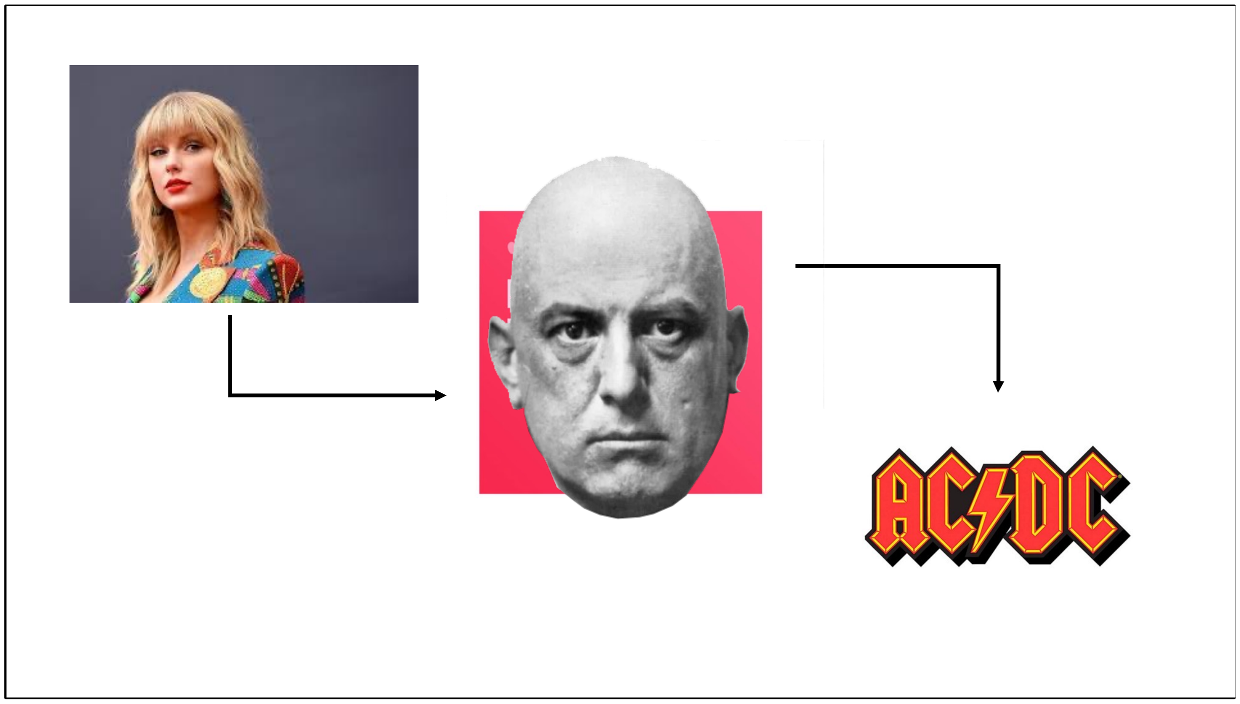


but what if you want it to recommend you something different, more out there?

the intuitive answer is that you should start listening to a lot of very different music,

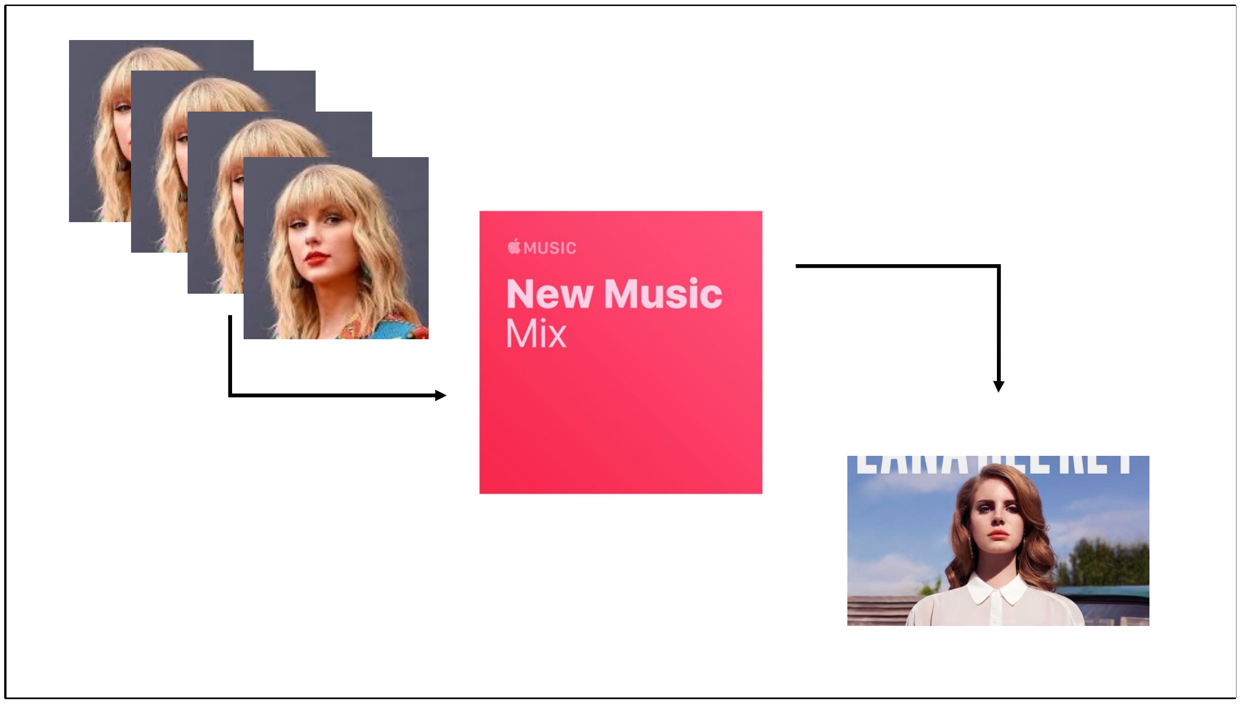


but that might not work — we don't really know what's going on behind the scenes. we just have to trust our gut instinct about it and hope it works.



alastair would be very proud.

by showing what's going on — giving people some insight into the rules, even if you don't give them _control_ over them — can help them leverage that algorithm even more.



maybe it turns out that it's not listening to different genres a lot in one sitting that triggers the algorithm to recommend me new stuff.

maybe it's changing the frequency — listening to it more times over the week, regardless of how long i do it.

maybe it's how often i skip tracks.

who knows! wouldn't it be nice if there was a way to find out?

FEEDBACK

WHAT YOU SHOW

and this is what i mean about revealing the right complexities.

the user doesn't need to know that the similarity matrix of two users can be used to generate an eigenvalue of confidence that the recommendations are relevant.

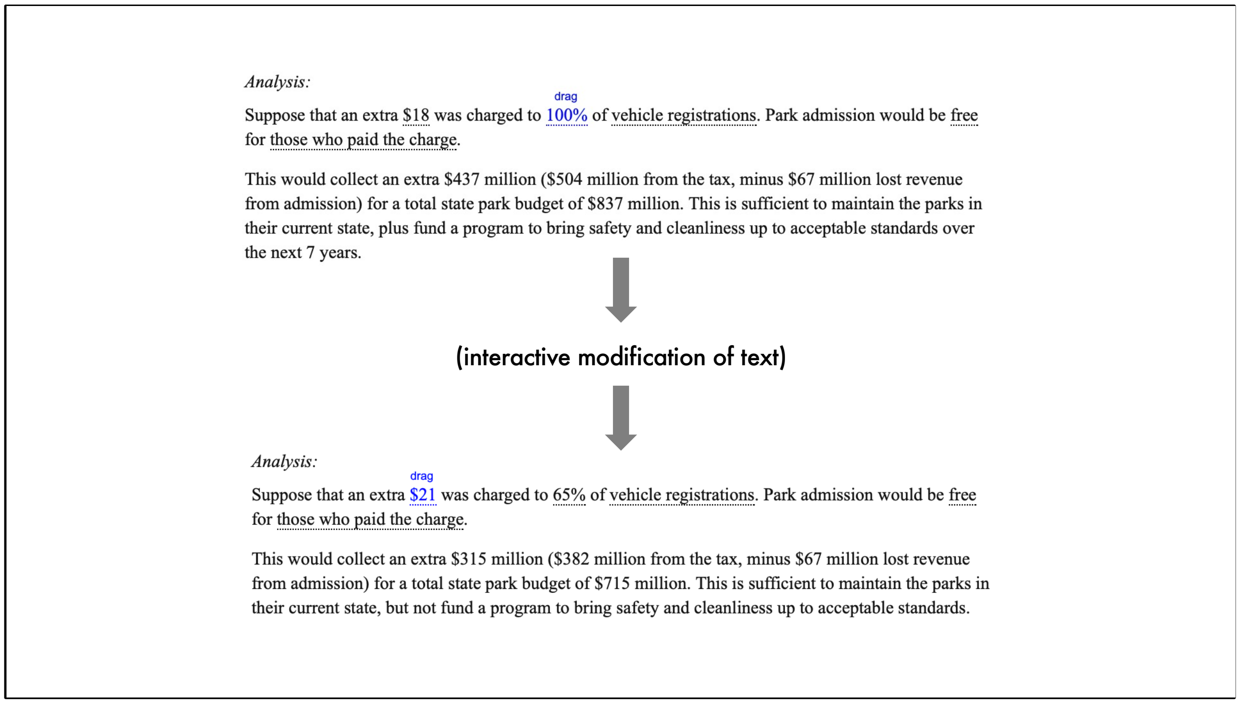
all they want to know is what they have to do to get the kind of recommendations they want.

if what we're trying to do is generate an answer space for our users, they'll naturally use feedback to do that.

but we can speed that up by explaining to them what the algorithm is doing.

Algorithms are tools

because at their core, algorithms are tools — remember that, because we're going to come back to it — and nobody wants to use a tool they don't understand.


Analysis:

drag

Suppose that an extra $18 was charged to 100% of vehicle registrations. Park admission would be free for those who paid the charge.

This would collect an extra $437 million ($504 million from the tax, minus $67 million lost revenue from admission) for a total state park budget of $837 million. This is sufficient to maintain the parks in their current state, plus fund a program to bring safety and cleanliness up to acceptable standards over the next 7 years.

(interactive modification of text)

Analysis:

drag

Suppose that an extra $21 was charged to 65% of vehicle registrations. Park admission would be free for those who paid the charge.

This would collect an extra $315 million ($382 million from the tax, minus $67 million lost revenue from admission) for a total state park budget of $715 million. This is sufficient to maintain the parks in their current state, but not fund a program to bring safety and cleanliness up to acceptable standards.

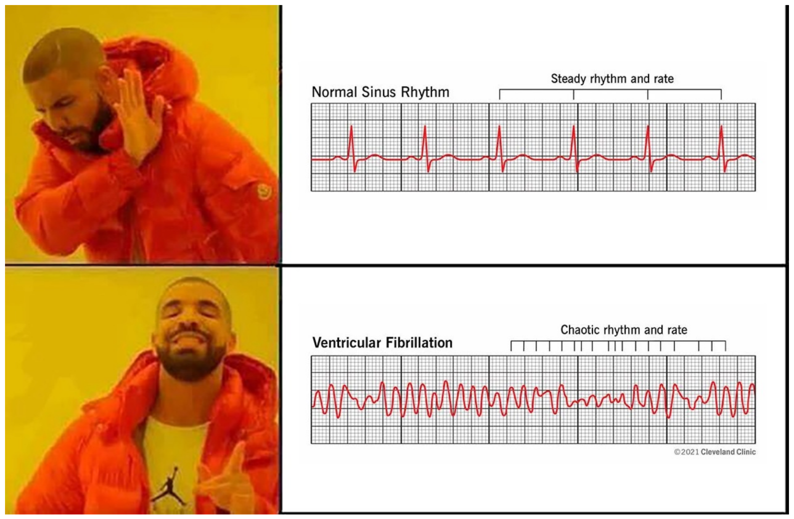
i think bret victor has a really nice way of approaching this. he has this concept of "explorable explanations" where you can enter in information and — in real time —

see what the output is going to be.

this is so simple but it is immensely powerful, and you can use this on your algorithms. think of it like allowing the user to run tiny simulations about things.

The user can modify the inputs and immediately see how the algorithm will understand that.

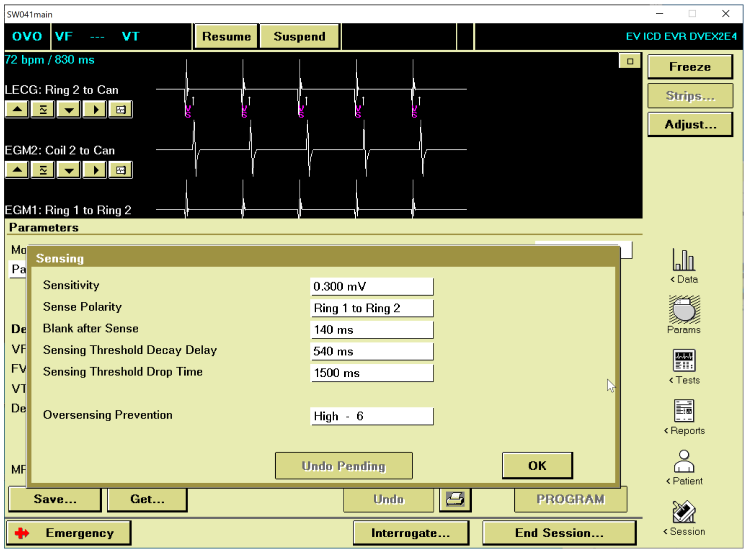
i work in hearts, so let me give you an example there.



so a key part of what a pacemaker has to do is look at the patient's ECG and make a determination:

is this normal and can I ignore it?

Or is it dangerous, and should I do something about it?

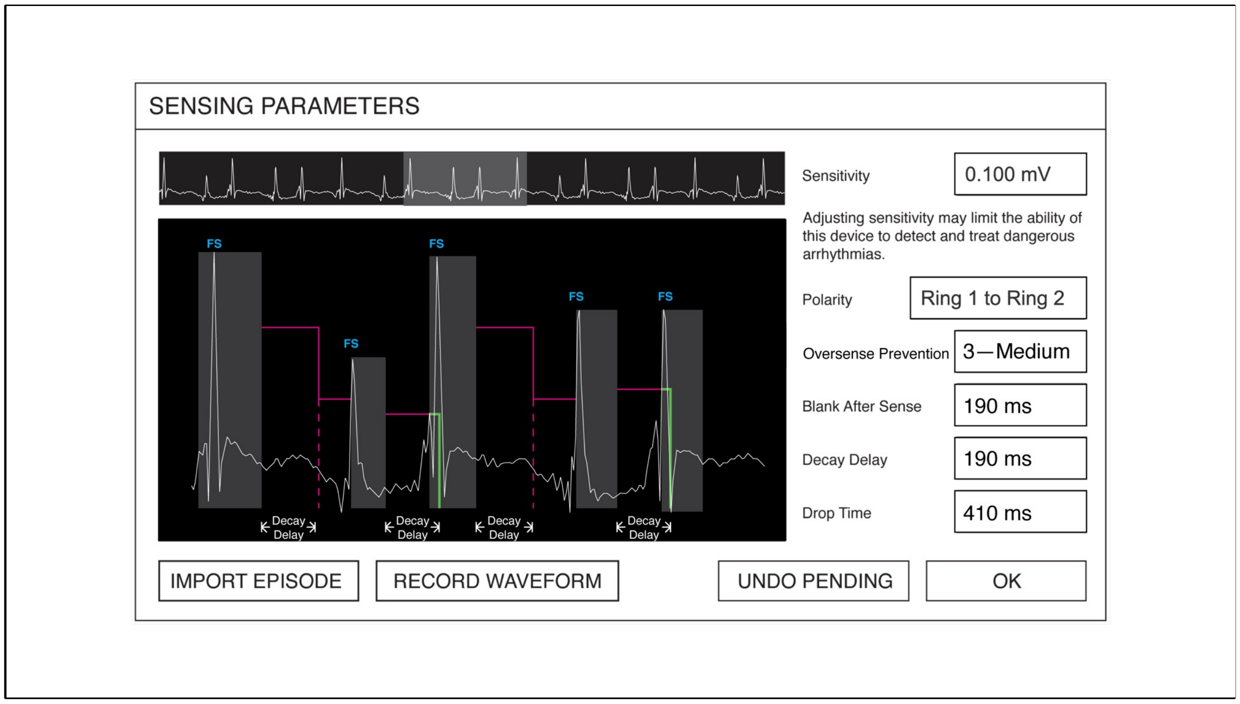


Sensing

Sensitivity	0.300 mV
Sense Polarity	Ring 1 to Ring 2
Blank after Sense	140 ms
Sensing Threshold Decay Delay	540 ms
Sensing Threshold Drop Time	1500 ms
Oversensing Prevention	High - 6

Now, you can give the user this list of parameters to adjust to help it with that detection. Tell the algorithm where and how to look for those dangerous arrhythmias.

But even though this gives you control over the algorithm, it gives you no *insight.* It’s really hard to make smart decisions without insight.



Sensitivity: 0.100 mV

Adjusting sensitivity may limit the ability of this device to detect and treat dangerous arrhythmias.

Polarity: Ring 1 to Ring 2

Oversense Prevention: 3—Medium

Blank After Sense: 190 ms

Decay Delay: 190 ms

Drop Time: 410 ms

So we built this system that allows the user to see how the pacemaker is going to interpret signals from the heart.

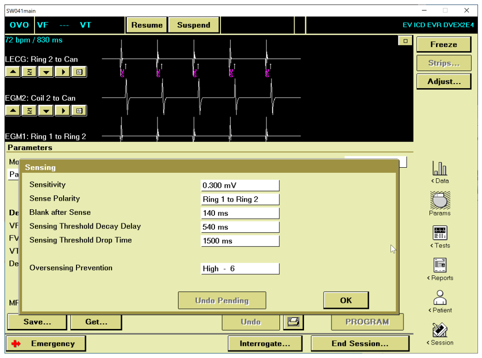
The algorithm are those pink lines there. Every time there's a spike, there's a period where the system isn't detecting anything.

That's those grey boxes.

By adjusting the parameters on the right, you could adjust the algorithm on the overlay and in real-time see how the device would interpret the signals.

This was really important because this was a brand-new algorithm for a brand-new class of devices.

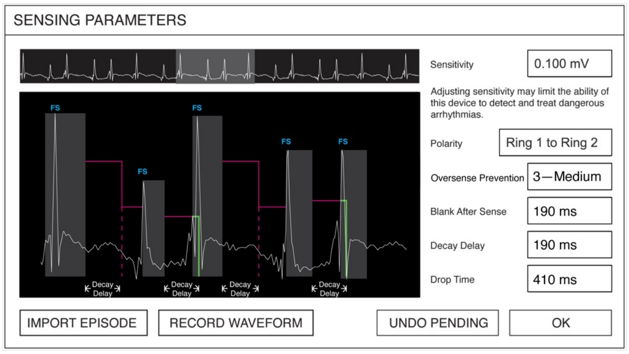
Nobody really knew how these worked, so we created this so they could experiment with them and feel confident in their understanding of how the algorithm worked.



Sensing

Sensitivity	0.300 mV
Sense Polarity	Ring 1 to Ring 2
Blank after Sense	140 ms
Sensing Threshold Decay Delay	540 ms
Sensing Threshold Drop Time	1500 ms
Oversensing Prevention	High - 6

Control without insight



SENSING PARAMETERS

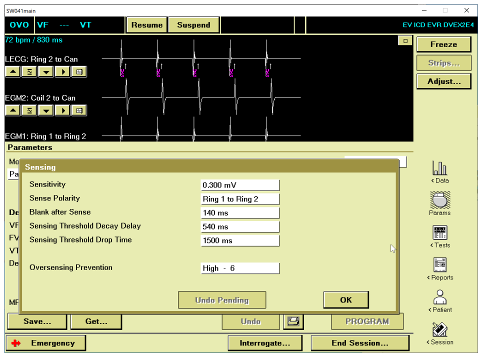
Sensitivity	0.100 mV
Polarity	Ring 1 to Ring 2
Oversense Prevention	3—Medium
Blank After Sense	190 ms
Decay Delay	190 ms
Drop Time	410 ms

Adjusting sensitivity may limit the ability of this device to detect and treat dangerous arrhythmias.

Control and insight

These are the exact same controls, but now you can immediately understand how your decisions will impact the system.

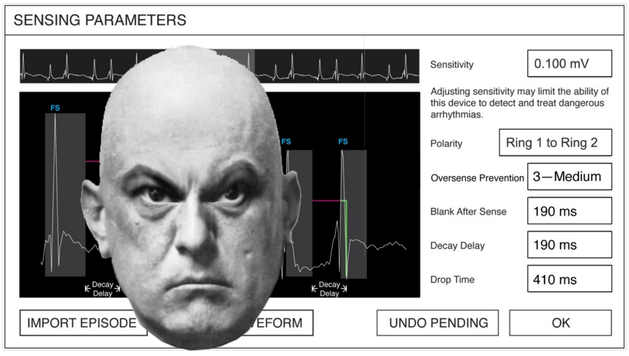
We didn't modify anything about how the algorithm works — all we did was *show* how it works.



Sensing

Sensitivity	0.300 mV
Sense Polarity	Ring 1 to Ring 2
Blank after Sense	140 ms
Sensing Threshold Decay Delay	540 ms
Sensing Threshold Drop Time	1500 ms
Oversensing Prevention	High - 6

Control without insight



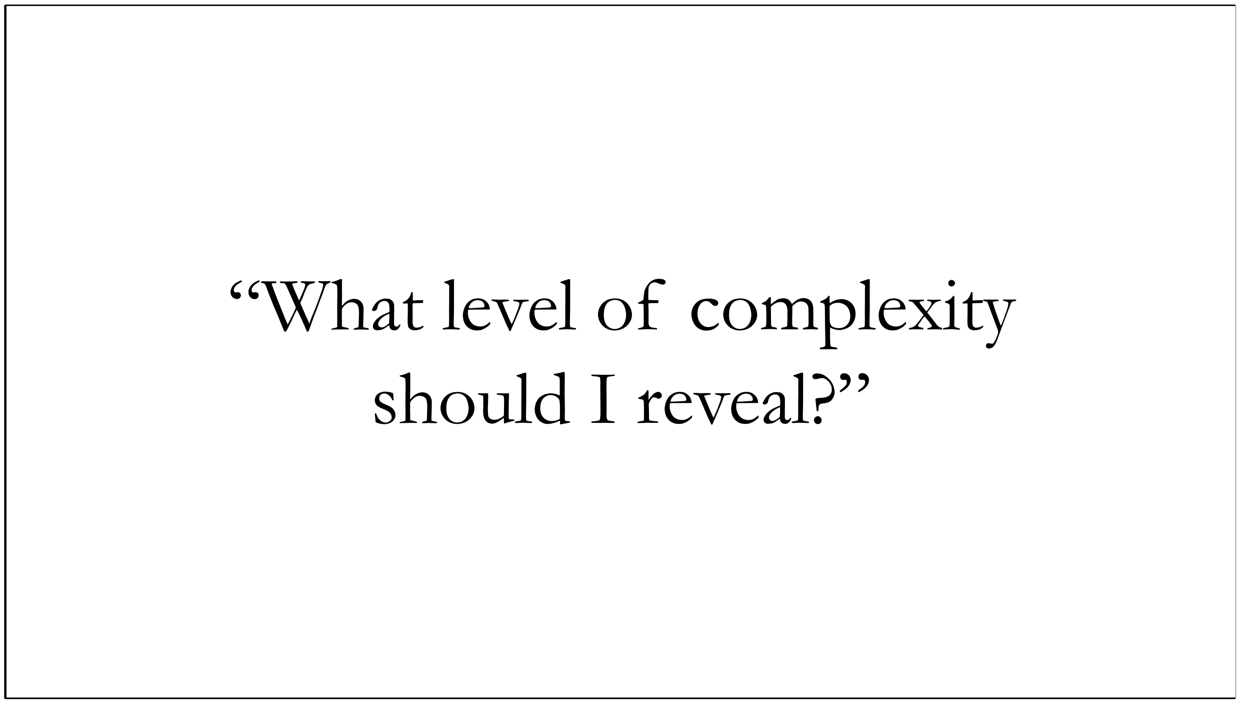
SENSING PARAMETERS

Sensitivity	0.100 mV
Polarity	Ring 1 to Ring 2
Oversense Prevention	3—Medium
Blank After Sense	190 ms
Decay Delay	190 ms
Drop Time	410 ms

Adjusting sensitivity may limit the ability of this device to detect and treat dangerous arrhythmias.

Control and insight

Aleister *hates* that.



“What level of complexity should I reveal?”

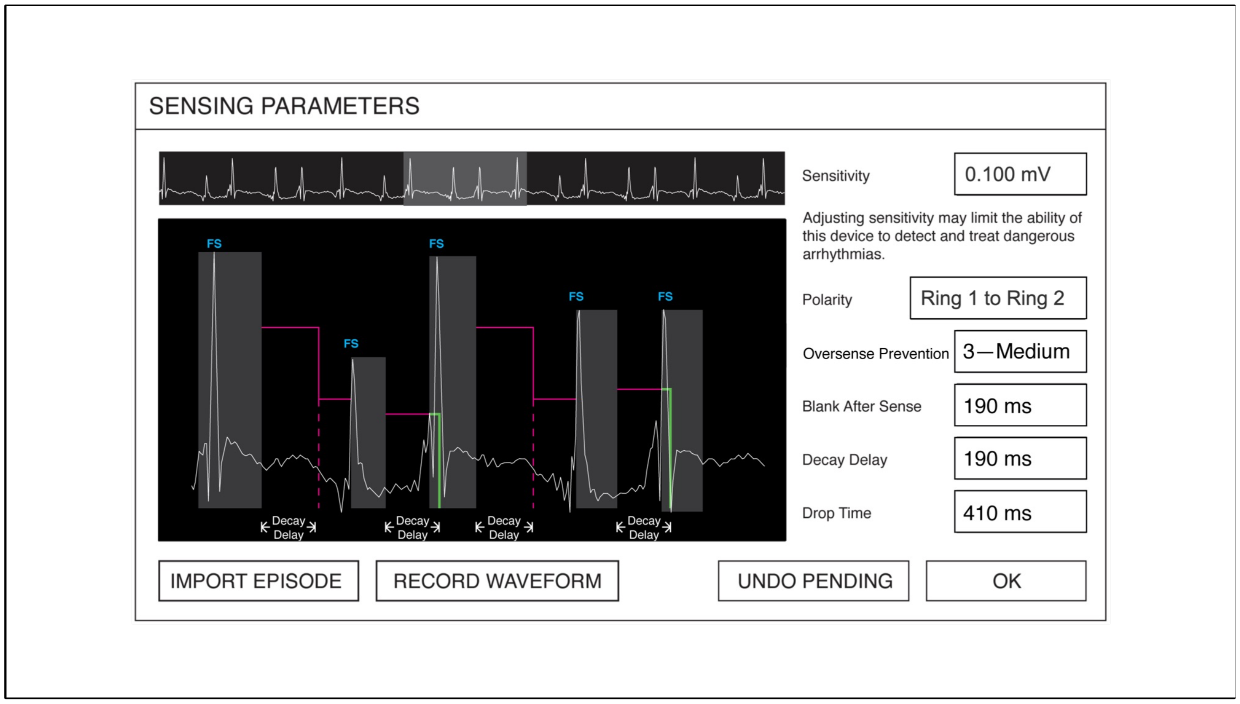
So now you may be wondering —

only you can know that.

you need to understand what it is that your users are trying to do with this tool that the algorithm is a part of,

what they need to do that job well, and what's going to get in their way.

go out and find out! do your user research on this, and test what you built!


SENSING PARAMETERS

Sensitivity: 0.100 mV

Adjusting sensitivity may limit the ability of this device to detect and treat dangerous arrhythmias.

Polarity: Ring 1 to Ring 2

Oversense Prevention: 3—Medium

Blank After Sense: 190 ms

Decay Delay: 190 ms

Drop Time: 410 ms

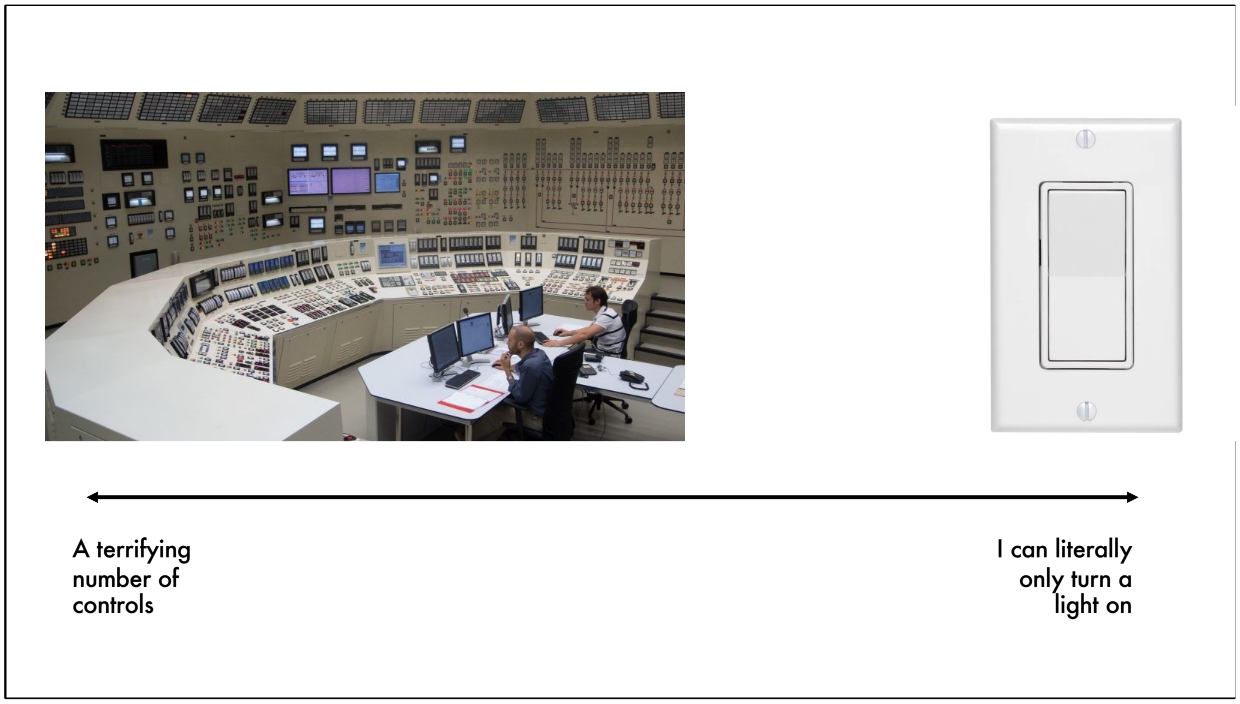
In this case, we learned that users were trying to get those FS markers to appear every time, and ONLY for the big spikes.

What they didn’t want to happen was have a bunch of noise come through and get interpreted as something dangerous.

The “IMPORT EPISODE” button down there let them bring in noise to test it out and see if their settings worked for these spikes, but not for noise.

so now we have feedback and these explorable explanations.

another really powerful way of explaining what an algorithm can do comes implicitly from the controls you reveal to your users.


A terrifying number of controls
I can literally only turn a light on

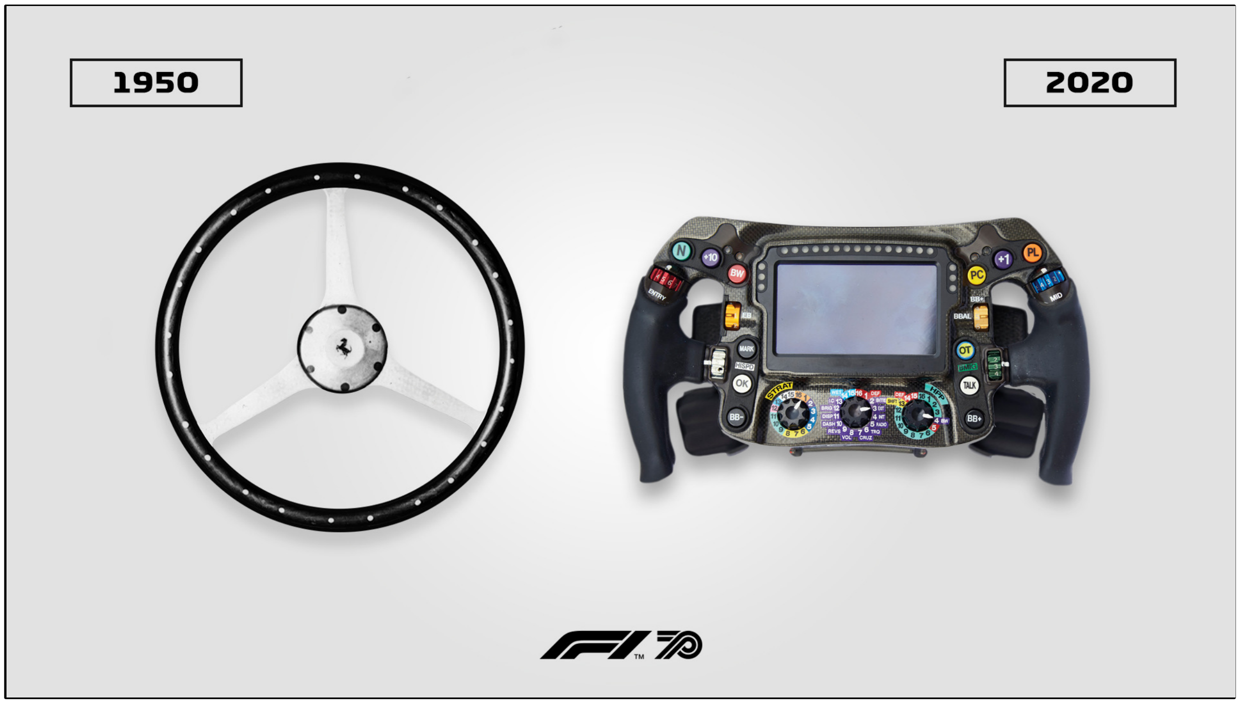
.typically you don't want users to control the entire nuclear facility, but you also want them to do more than flick a switch

most algorithms need to be adapted and given some control over how they work.

and it gets even more complex — because your users are likely to have a huge range of skill and knowledge levels.

some users might _need_ more control and more access to the inner workings of an algorithm,

but it's probably bad to let a novice get in that deep.



The things you show and hide give the user an understanding of the space they have to operate in and what the algorithm will show them.

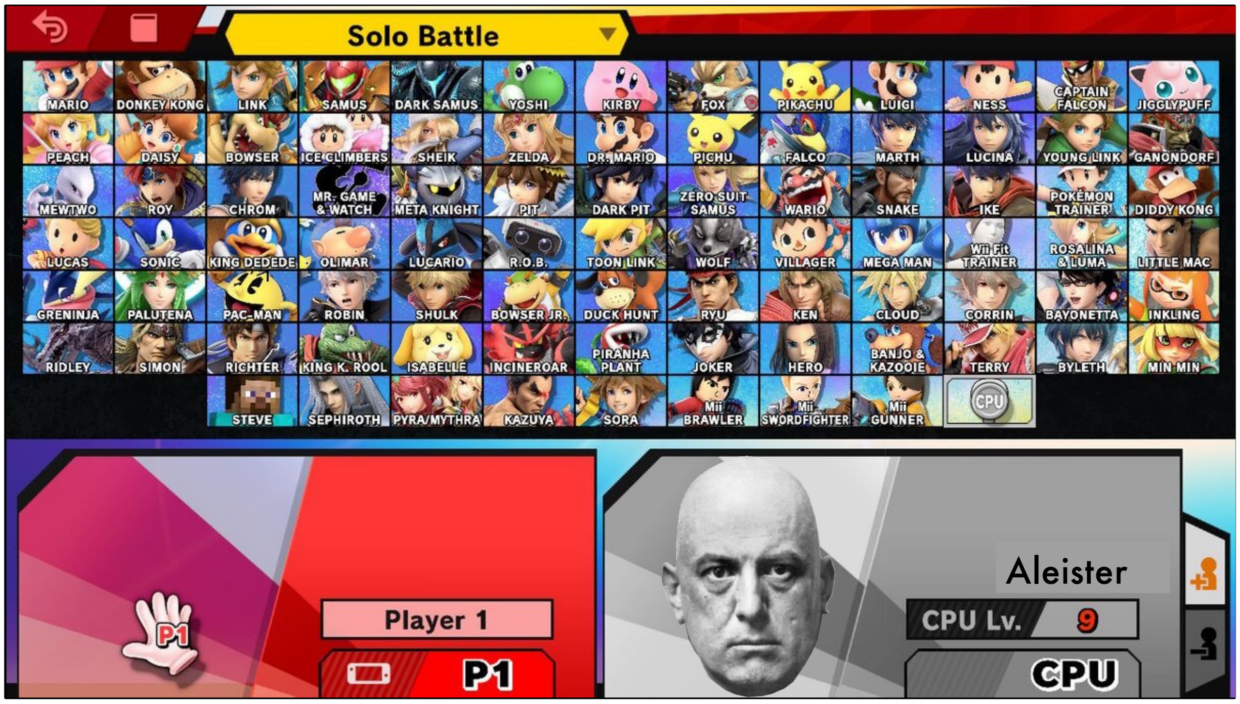
These two steering wheels give you radically different insights into the vehicle.

One lets you turn the car, the other lets you do a lot more.

They give you immediate, implicit understanding of the complexity of what you’re working with.

The parameters — the controls — you give someone access to so they can tweak an algorithm helps them understand what it does.

Complex systems have more controls.



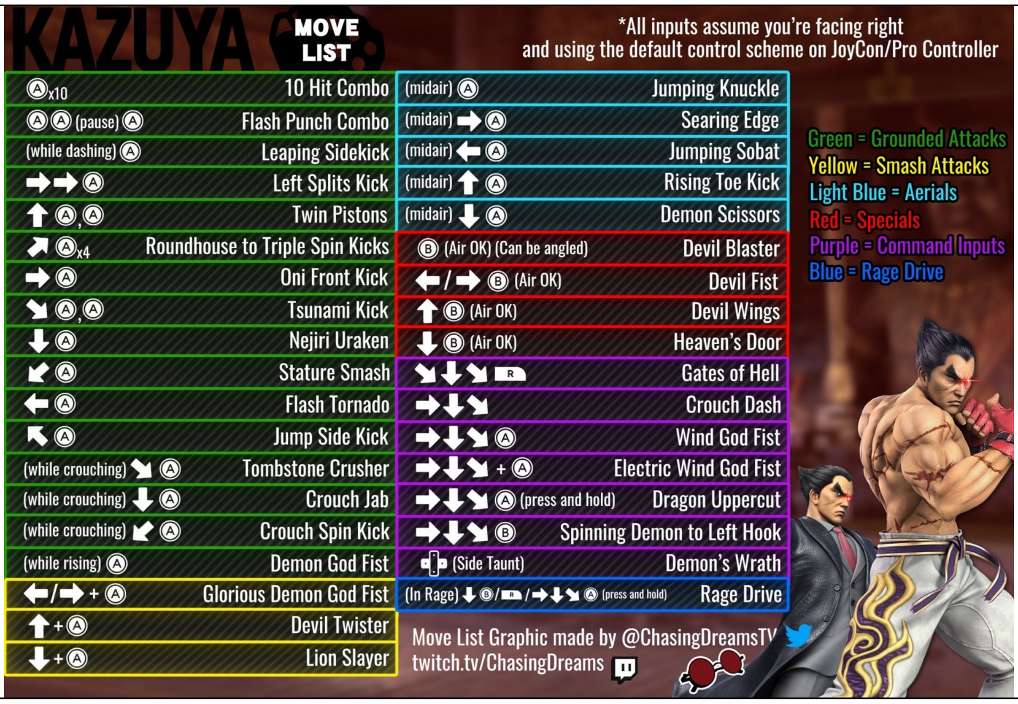
And you have to show some of those controls to your users so they understand what they can do

You ever been in a situation where you're hanging out with your friends and they're like "hey let's play this game"

and already it's overwhelming

And then they sit you down and the controller has like, WAY too many buttons

And you ask “so what am I supposed to do?”


KAZUYA MOVE LIST

*All inputs assume you're facing right and using the default control scheme on JoyCon/Pro Controller

Ⓐx10	10 Hit Combo
Ⓐ Ⓐ (pause) Ⓐ	Flash Punch Combo
(while dashing) Ⓐ	Leaping Sidekick
→ → Ⓐ	Left Splits Kick
↑ Ⓐ, Ⓐ	Twin Pistons
↗ Ⓐx4	Roundhouse to Triple Spin Kicks
→ Ⓐ	Oni Front Kick
↘ Ⓐ, Ⓐ	Tsunami Kick
↓ Ⓐ	Nejiri Uraken
↙ Ⓐ	Stature Smash
← Ⓐ	Flash Tornado
↖ Ⓐ	Jump Side Kick
(while crouching) ↘ Ⓐ	Tombstone Crusher
(while crouching) ↓ Ⓐ	Crouch Jab
(while crouching) ↙ Ⓐ	Crouch Spin Kick
(while rising) Ⓐ	Demon God Fist
←/→ + Ⓐ	Glorious Demon God Fist
↑ + Ⓐ	Devil Twister
↓ + Ⓐ	Lion Slayer
(midair) Ⓐ	Jumping Knuckle
(midair) → Ⓐ	Searing Edge
(midair) ← Ⓐ	Jumping Sobat
(midair) ↑ Ⓐ	Rising Toe Kick
(midair) ↓ Ⓐ	Demon Scissors
Ⓑ (Air OK) (Can be angled)	Devil Blaster
←/→ Ⓑ (Air OK)	Devil Fist
↑ Ⓑ (Air OK)	Devil Wings
↓ Ⓑ (Air OK)	Heaven's Door
↘ ↓ ↘ R	Gates of Hell
→ ↓ ↘	Crouch Dash
→ ↓ ↘ Ⓐ	Wind God Fist
→ ↓ ↘ + Ⓐ	Electric Wind God Fist
→ ↓ ↘ Ⓐ (press and hold)	Dragon Uppercut
→ ↓ ↘ Ⓑ	Spinning Demon to Left Hook
(Side Taunt)	Demon's Wrath
(In Rage) ↓ Ⓑ/R/→ ↓ ↘ Ⓐ (press and hold)	Rage Drive

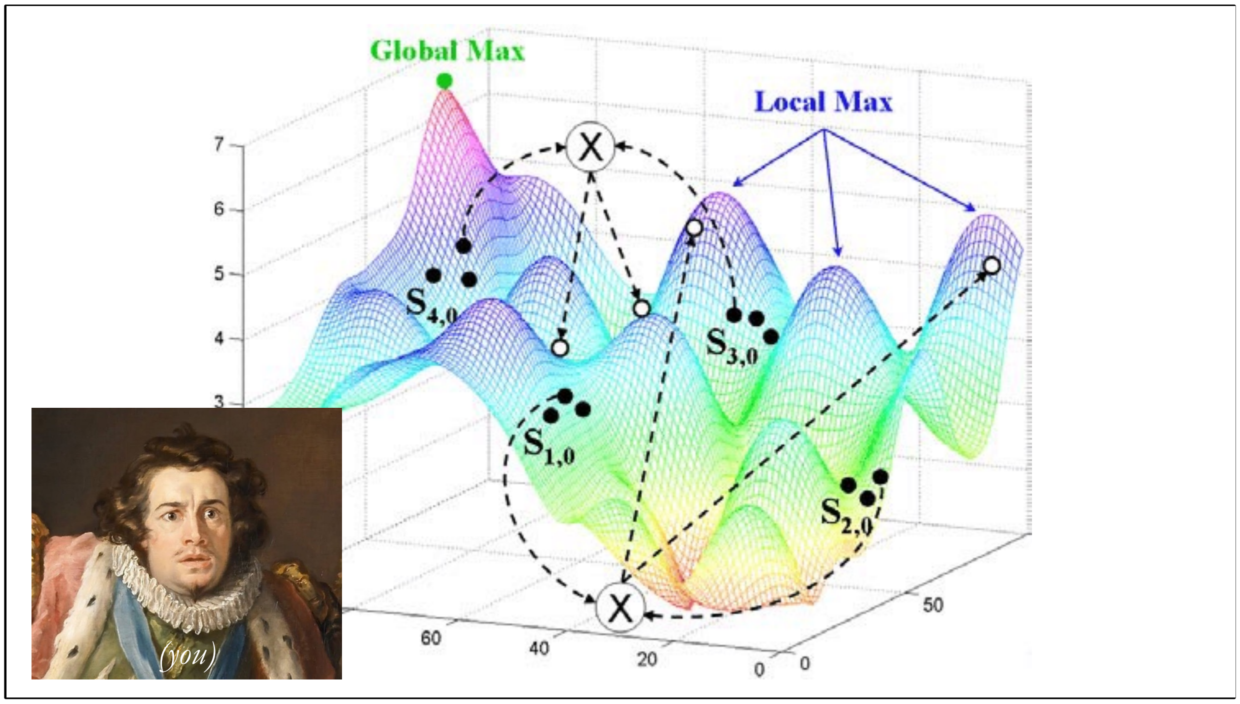
Green = Grounded Attacks
Yellow = Smash Attacks
Light Blue = Aerials
Red = Specials
Purple = Command Inputs
Blue = Rage Drive

Move List Graphic made by @ChasingDreamsTV
twitch.tv/ChasingDreams

And they're like — okay here are the controls, just try to fight us

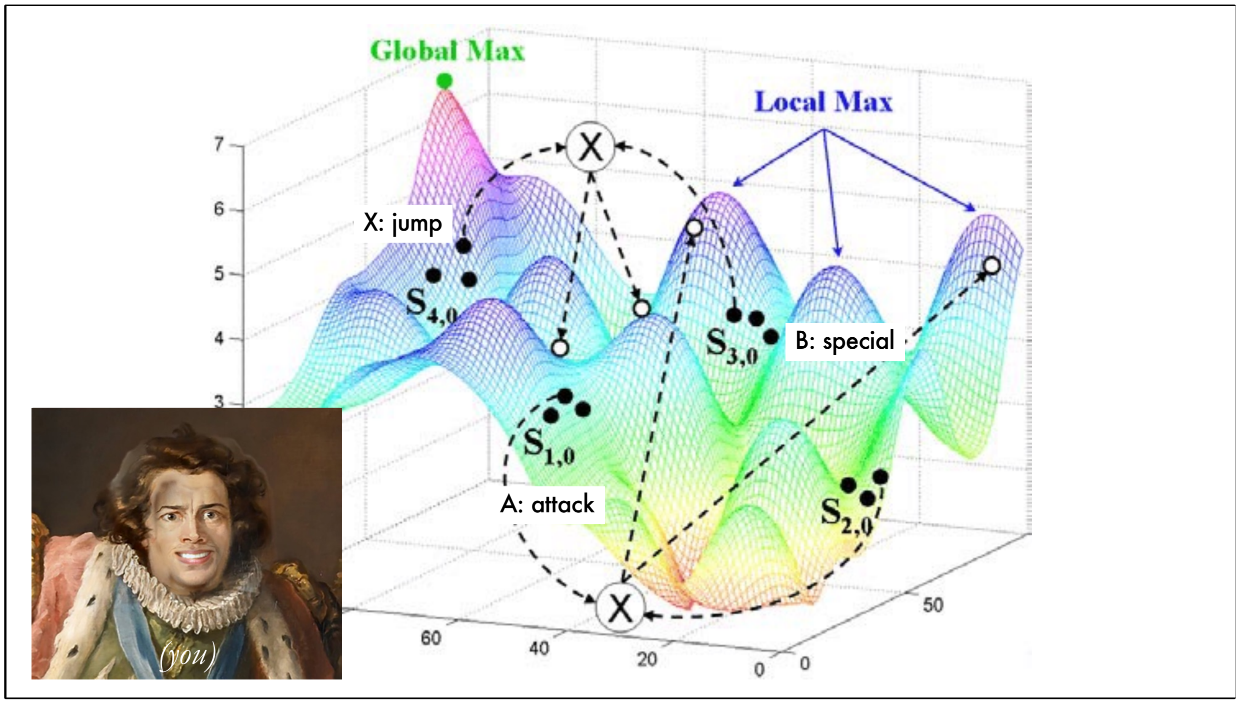
And you're like "ooooohkaaaaay"

And then you start playing and they just absolutely kick your ass because they know everything about it and you don’t?



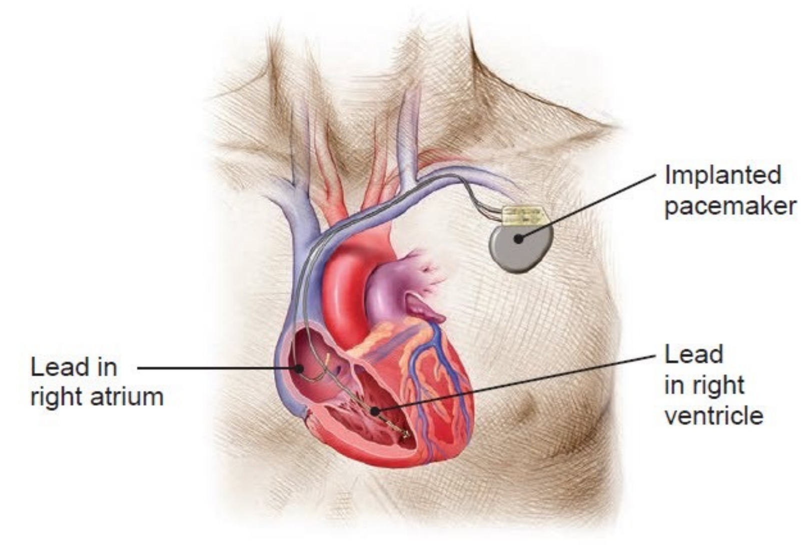
That’s because your space is way, way too big to operate in.

But if you have a few handles on what you can do and what the goal is…



Well all of a sudden you feel a lot better about things.

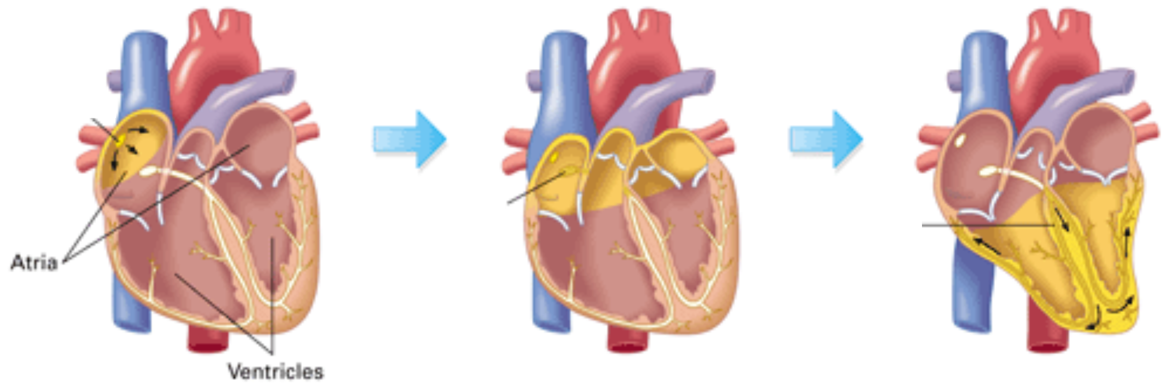
Exposing controls to the user gives them critical clues about what the algorithm is doing behind the scenes and how they can influence it.



i mentioned dangerous arrhythmias before — let's dig a little deeper.

see, most pacemakers have leads in two chambers of the heart.

the atria up here suck in blood and push it down into the ventricles, which then push it out to the rest of the body.



that suck-push sequence is very precisely timed — that lub-dub you feel in your heartbeat is your atria contracting, then your ventricles contracting.

turns out that if you get that sequence wrong — you contract the ventricle too fast after the atrium for example — people's health really degrades.

it's very important to get that timing bang on.

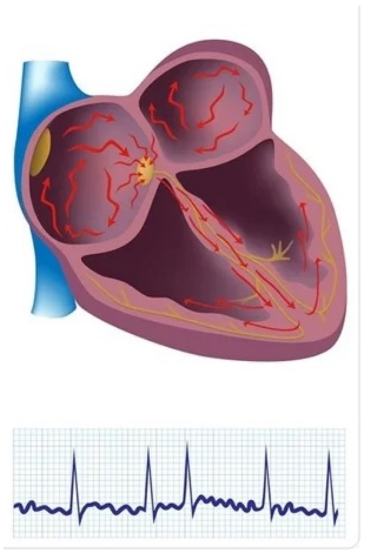
a very common issue is called AV block —

basically, the atria will beat and the ventricles will beat, they just don't do it at the same time.

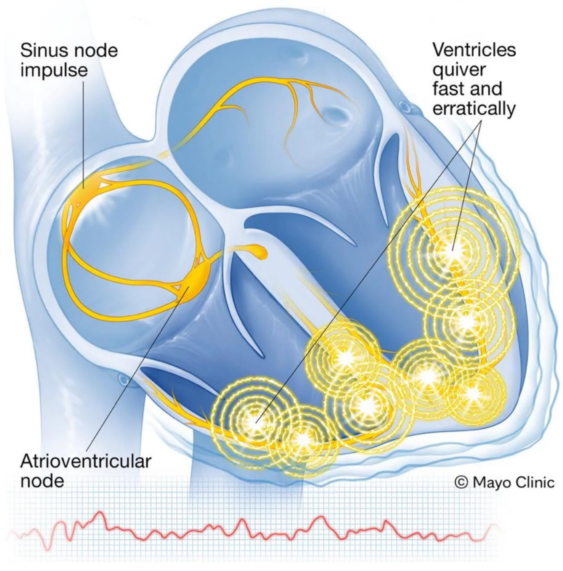
to fix this, you put a lead in the atrium that listens for the beat, and then a lead in the ventricle that paces that chamber.

so the atrium contracts, and the pacemaker forces the ventricle to contract.

Atrial Fibrillation
(bad for you)

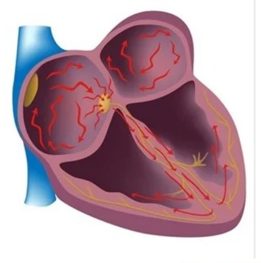


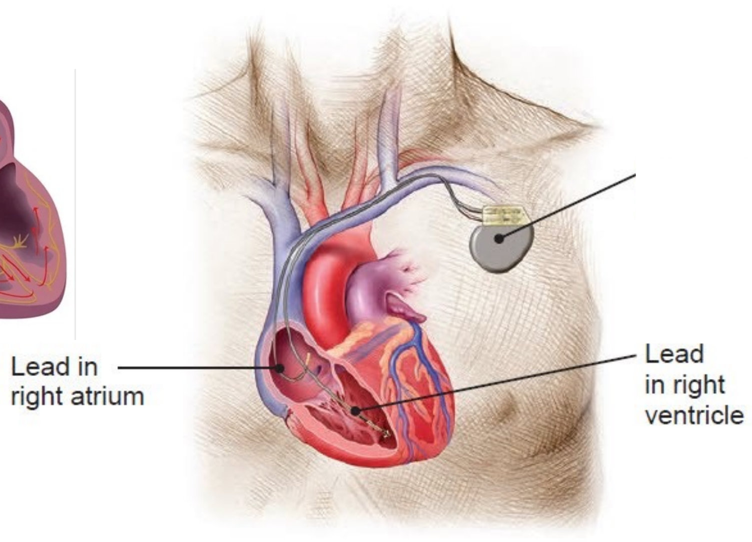
Ventricular Fibrillation
(EXTREMELY BAD FOR YOU)



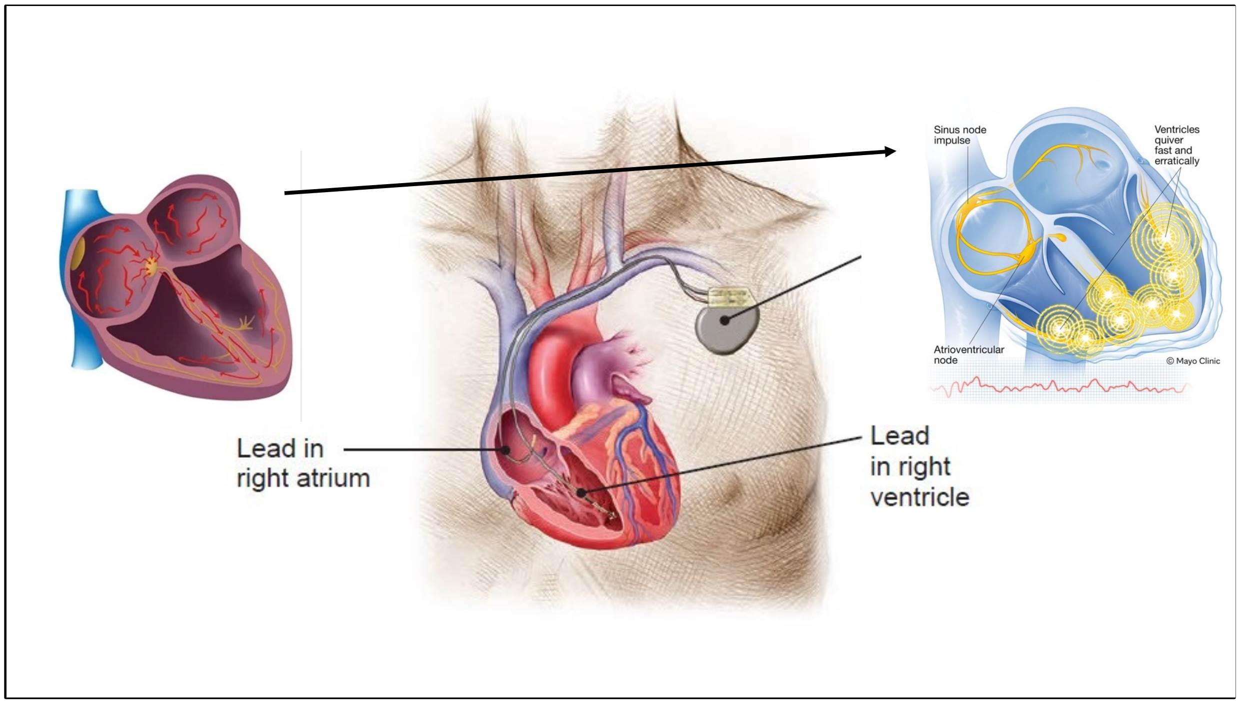
this works really well! but hearts can be really unpredictable — and sometimes the atrium will beat in a dangerous way,

like way too quickly. this is called fibrillation, and in the atrium it's bad. but in the ventricle, it's fatal within minutes.





so if our pacemaker is listening to the atrium and pacing the ventricle at the same time, we have a problem if the atrium starts fibrillating.



the pacemaker will happily beat the ventricle every time the atrium does — including sending those fibrillations down to the ventricle.

boom, you have a dead patient.

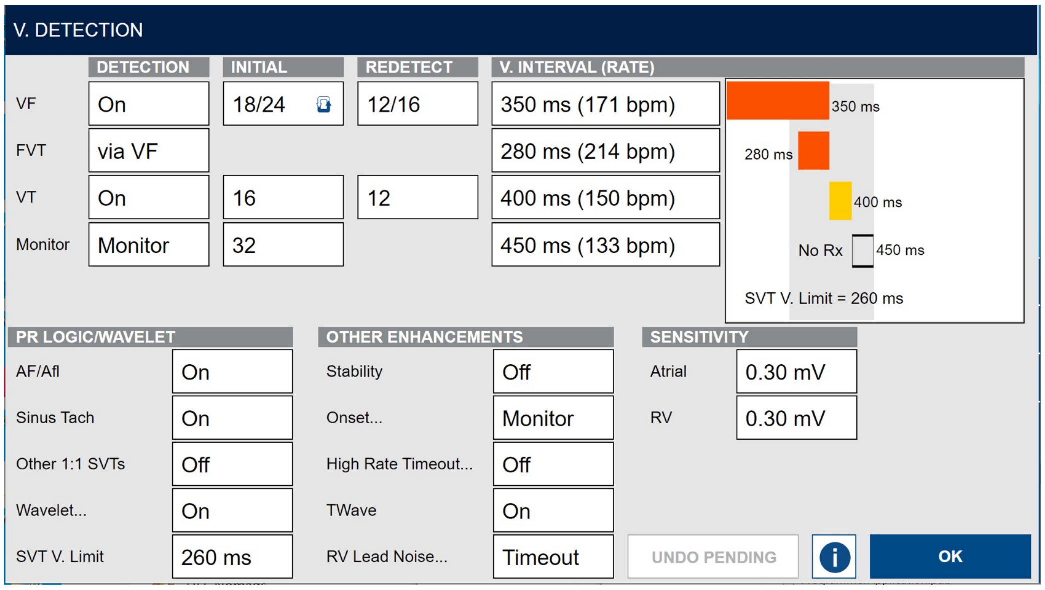
so pacemakers have a lot of very complex algorithms in them that are used to respond correctly to those dangerous arrhythmias.

and they work pretty well out of the box, but everyone is different, so we need to give control over them to our users.

some patients need very little intervention, and some patients are really complex and require a _lot_ of tweaking.

and some of our users are experts with decades of experience working with these devices, and others are total novices.

what do we do?



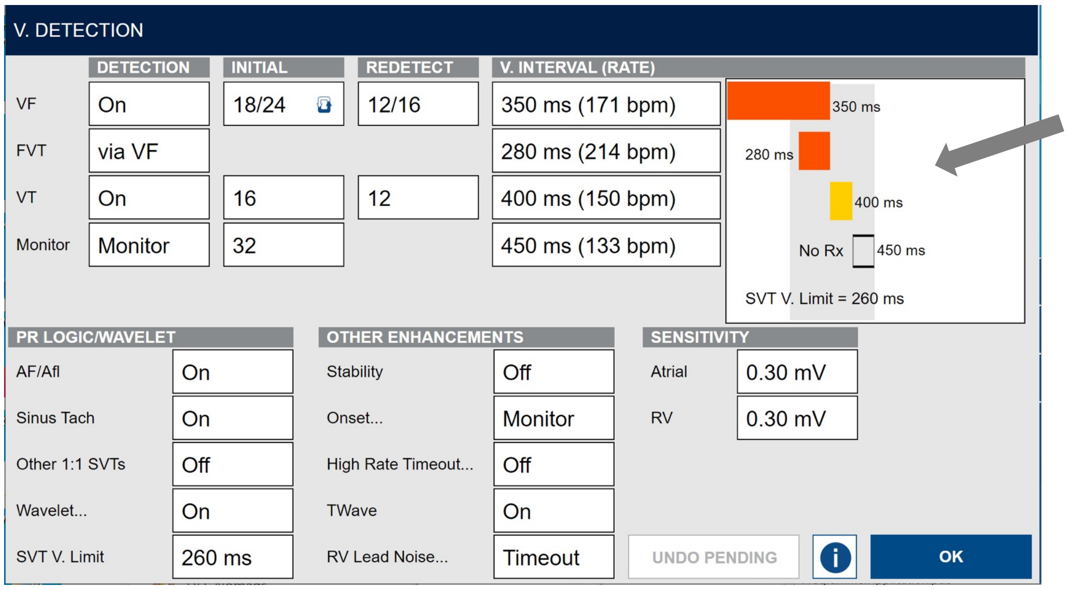
V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	18/24	12/16	350 ms (171 bpm)
FVT	via VF			280 ms (214 bpm)
VT	On	16	12	400 ms (150 bpm)
Monitor	Monitor	32		450 ms (133 bpm)

PR LOGIC/WAVELET		OTHER ENHANCEMENTS		SENSITIVITY	
AF/Afl	On	Stability	Off	Atrial	0.30 mV
Sinus Tach	On	Onset...	Monitor	RV	0.30 mV
Other 1:1 SVTs	Off	High Rate Timeout...	Off		
Wavelet...	On	TWave	On		
SVT V. Limit	260 ms	RV Lead Noise...	Timeout		

the thing we _don't_ do is expose everything, even if our power users constantly ask for it.

instead, we basically make it hard to get to the complex stuff that few people need and really easy to get to the stuff that most people need a lot.

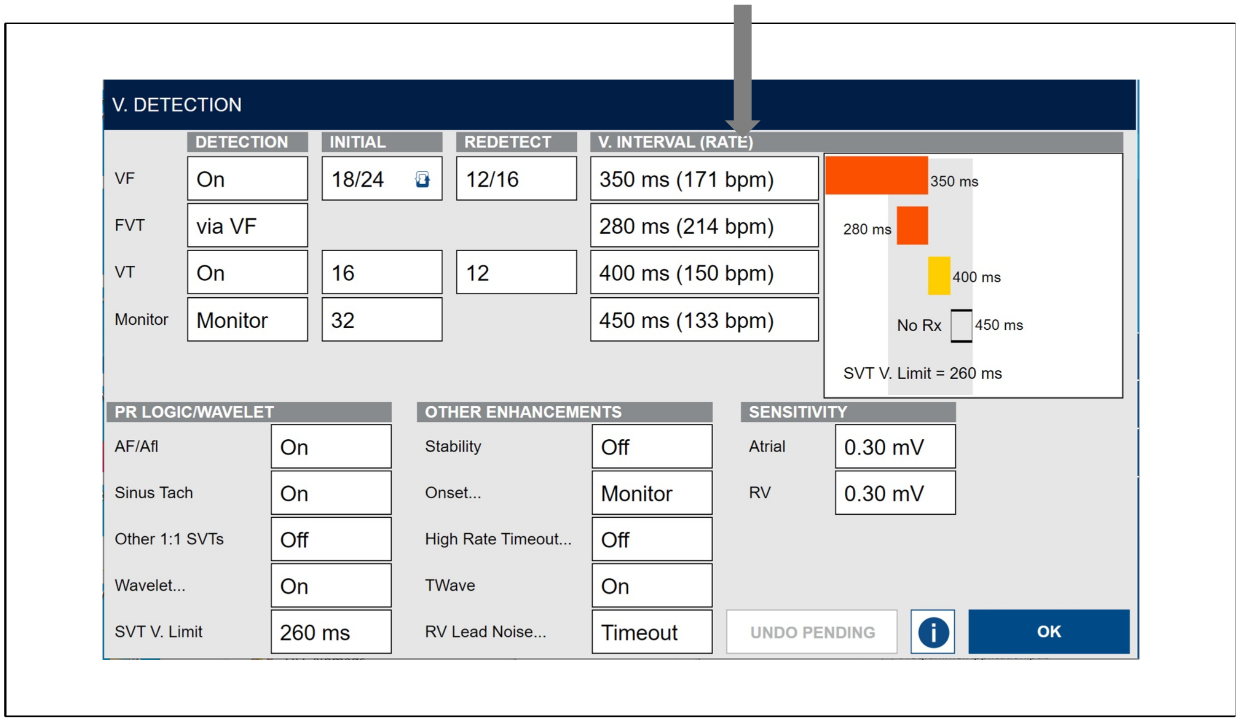


V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	18/24	12/16	350 ms (171 bpm)
FVT	via VF			280 ms (214 bpm)
VT	On	16	12	400 ms (150 bpm)
Monitor	Monitor	32		450 ms (133 bpm)

PR LOGIC/WAVELET		OTHER ENHANCEMENTS		SENSITIVITY	
AF/Afl	On	Stability	Off	Atrial	0.30 mV
Sinus Tach	On	Onset...	Monitor	RV	0.30 mV
Other 1:1 SVTs	Off	High Rate Timeout...	Off		
Wavelet...	On	TWave	On		
SVT V. Limit	260 ms	RV Lead Noise...	Timeout		

a lot of our users only ever need to adjust these rate bands. and these are a great example of explorable explanations! let's walk through this screen a little bit.



V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	18/24	12/16	350 ms (171 bpm)
FVT	via VF			280 ms (214 bpm)
VT	On	16	12	400 ms (150 bpm)
Monitor	Monitor	32		450 ms (133 bpm)

PR LOGIC/WAVELET		OTHER ENHANCEMENTS		SENSITIVITY	
AF/Afl	On	Stability	Off	Atrial	0.30 mV
Sinus Tach	On	Onset...	Monitor	RV	0.30 mV
Other 1:1 SVTs	Off	High Rate Timeout...	Off		
Wavelet...	On	TWave	On		
SVT V. Limit	260 ms	RV Lead Noise...	Timeout		

the key metric here is the heart rate — generally the faster the rate, the worse off our patient is.

but a really, really fast heart has different treatment needs than a slower one, so we give our users three different ways of managing them.

a moderately fast heart — that's ventricular tachycardia, or VT — is controlled from here. A little faster, that's Fast VT.

a very fast heart — ventricular fibrillation, VF — that's here.

they're color-coded and you can see how they all line up. you can see what rates are going to be covered and if there are any gaps. And if you adjust anything, it gets updated here too.

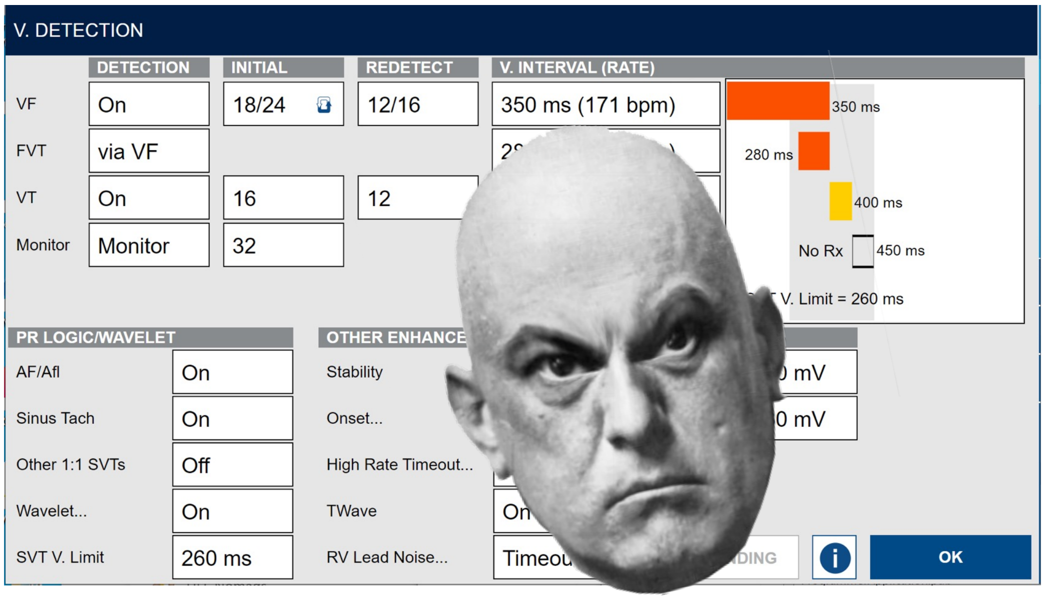
for most people, this is enough!

this gives them just enough information about the algorithm to understand how it's going to respond to different rates, and enough control to adjust it to suit their patient's needs.

and we did it all with just a little graph!

now our users don't have to just _faith_ that their patients will get the right treatments at the right time, they can actually see it happening.

these controls help demonstrate the space that a user can work in.



V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	18/24	12/16	350 ms (171 bpm)
FVT	via VF			2
VT	On	16	12	
Monitor	Monitor	32		

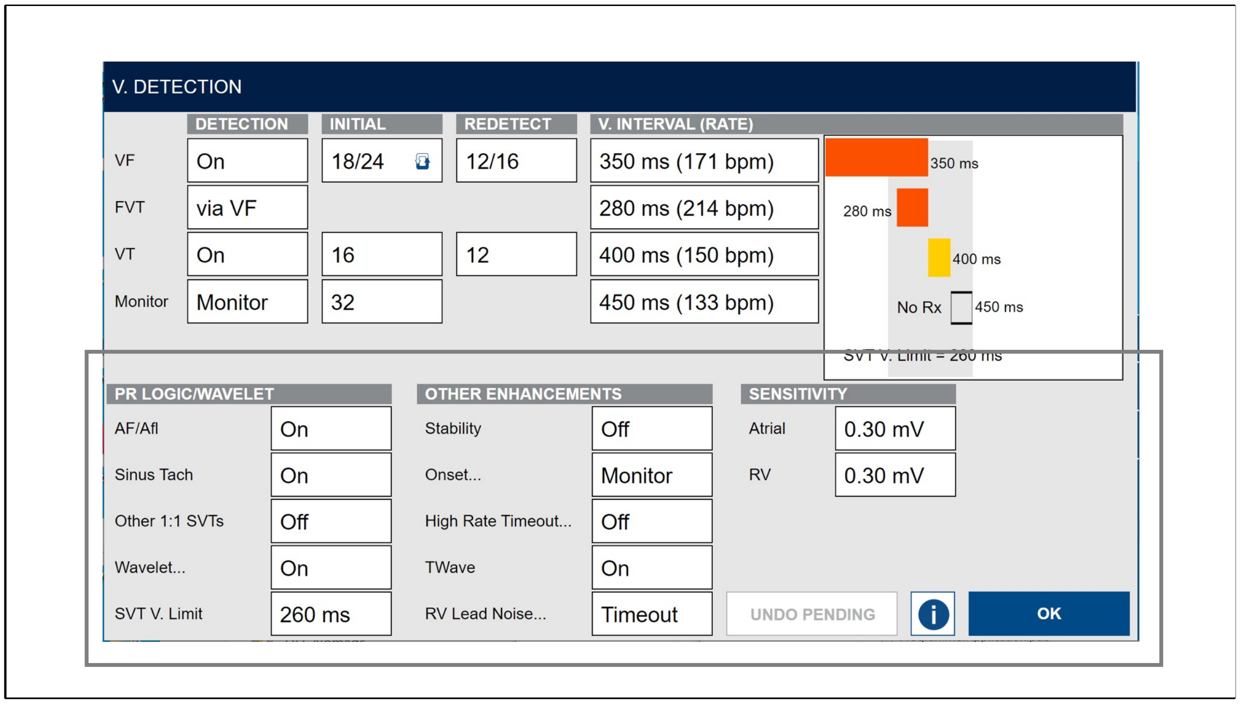
PR LOGIC/WAVELET	
AF/Afl	On
Sinus Tach	On
Other 1:1 SVTs	Off
Wavelet...	On
SVT V. Limit	260 ms

OTHER ENHANCE	
Stability	
Onset...	
High Rate Timeout...	
TWave	On
RV Lead Noise...	Timeou

0 mV

0 mV

alastair would not like this at all.



V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	18/24	12/16	350 ms (171 bpm)
FVT	via VF			280 ms (214 bpm)
VT	On	16	12	400 ms (150 bpm)
Monitor	Monitor	32		450 ms (133 bpm)

PR LOGIC/WAVELET		OTHER ENHANCEMENTS		SENSITIVITY	
AF/Afl	On	Stability	Off	Atrial	0.30 mV
Sinus Tach	On	Onset...	Monitor	RV	0.30 mV
Other 1:1 SVTs	Off	High Rate Timeout...	Off		
Wavelet...	On	TWave	On		
SVT V. Limit	260 ms	RV Lead Noise...	Timeout		

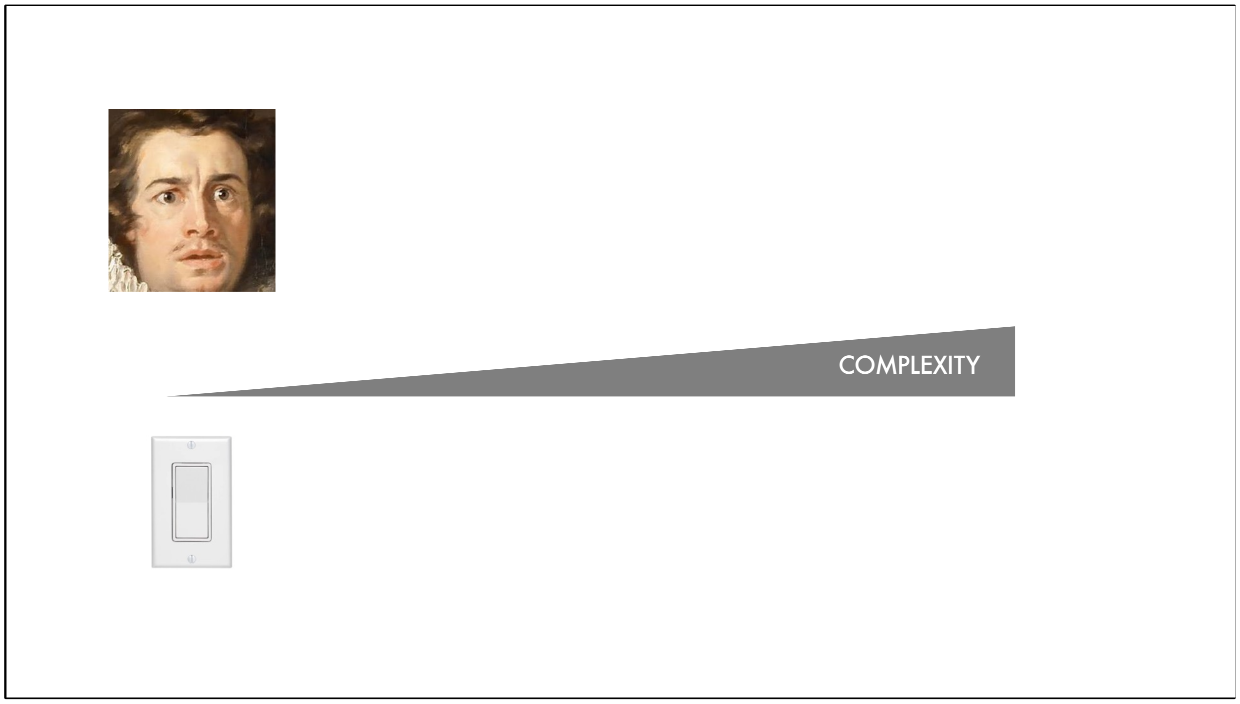
but some users need more control and have the knowledge to go with it.

so we give them the option, but they have to go digging for it.

if you click down here, you can see all the things you can do with the algorithm.

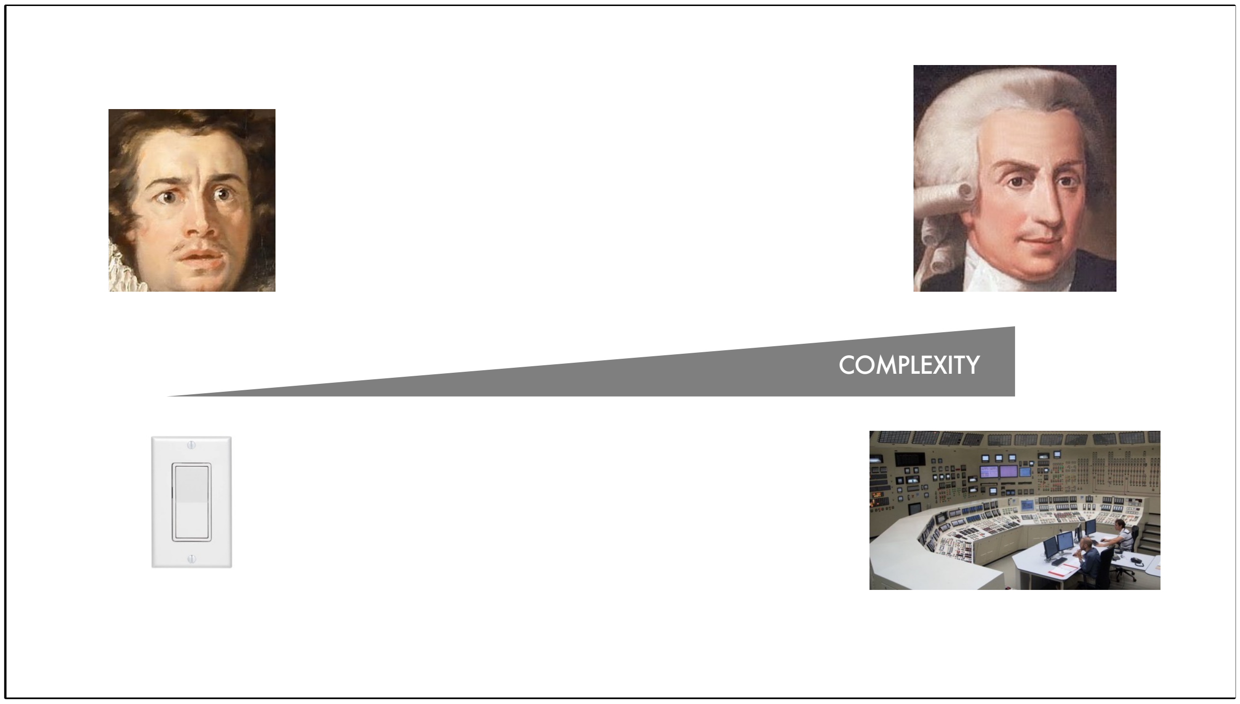
it gives a lot more control, and expands the possibility space.

the idea here is to give progressively more options to users who are looking for them, and protect the ones who aren't from accidentally doing something dangerous.



this _progressive complexity_ is really handy because it matches our users' own experiences.

when they start out, they don't know anything and struggle with light switches.



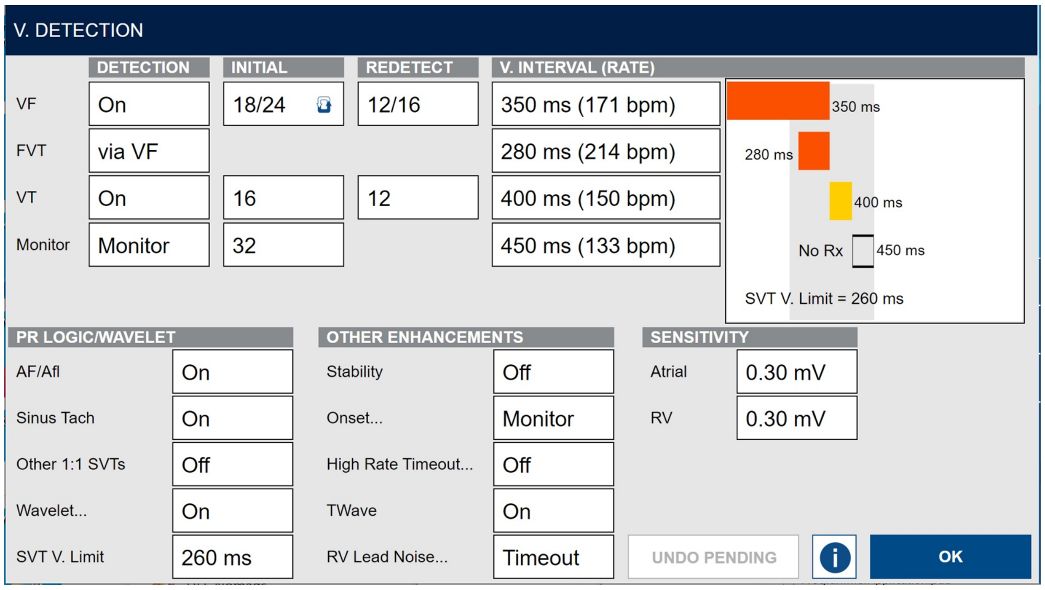
but as they get more experienced, they need that extra control, and when they go and look for it, we give it to them.

our users trust our system because it works consistently —

the feedback they see is that dangerous things are not happening, and efficacy of care is.

they have enough levers to pull that they can control the system to the level they need, based on their patient and their own knowledge.

and they can see how changes they make will impact their patients in advance.


V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	18/24	12/16	350 ms (171 bpm)
FVT	via VF			280 ms (214 bpm)
VT	On	16	12	400 ms (150 bpm)
Monitor	Monitor	32		450 ms (133 bpm)

PR LOGIC/WAVELET		OTHER ENHANCEMENTS		SENSITIVITY	
AF/Afl	On	Stability	Off	Atrial	0.30 mV
Sinus Tach	On	Onset...	Monitor	RV	0.30 mV
Other 1:1 SVTs	Off	High Rate Timeout...	Off		
Wavelet...	On	TWave	On		
SVT V. Limit	260 ms	RV Lead Noise...	Timeout		

what we've done here is more than just build trust in an algorithm, we've actually made our users _better at their jobs._

the UI — and the algorithm behind it — have enhanced their capability.

and this is really important, because we're not just building trust that an algorithm works.

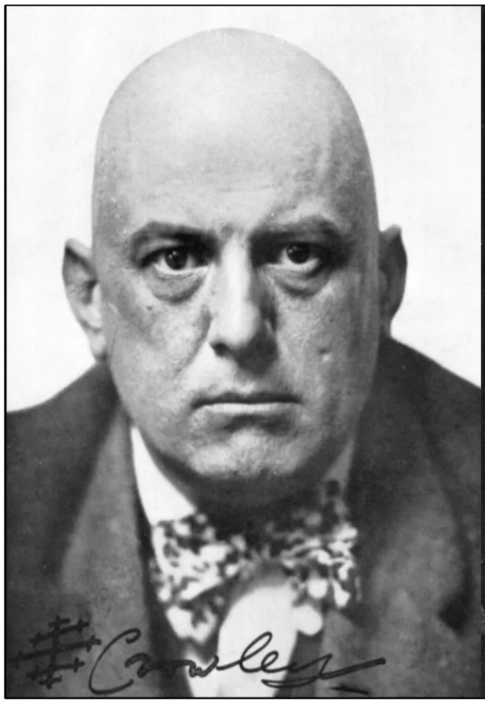
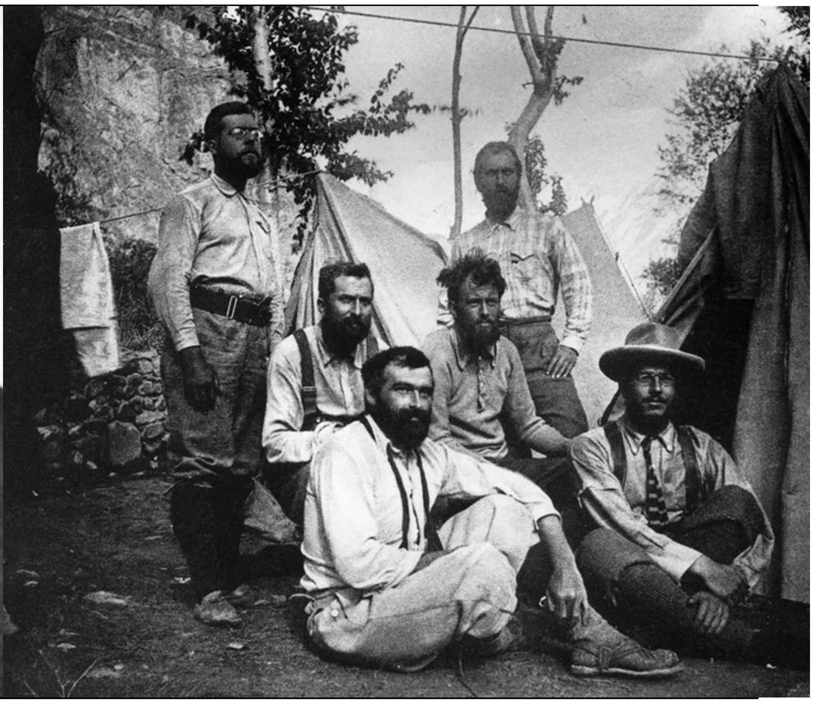
we also have to build trust that it's a _good thing._

all this talk about LLMs has a lot of people worried about what's going to happen to their jobs.

i think it's good to be concerned about this! if we don't think about these things — if we just blindly accept them into our lives — we don't really know the outcomes could be.

maybe it'll be a good thing,

or maybe we’ll ask everyone to take things on faith and wind up inventing a weird cult and letting all our buddies die on a mountaintop.

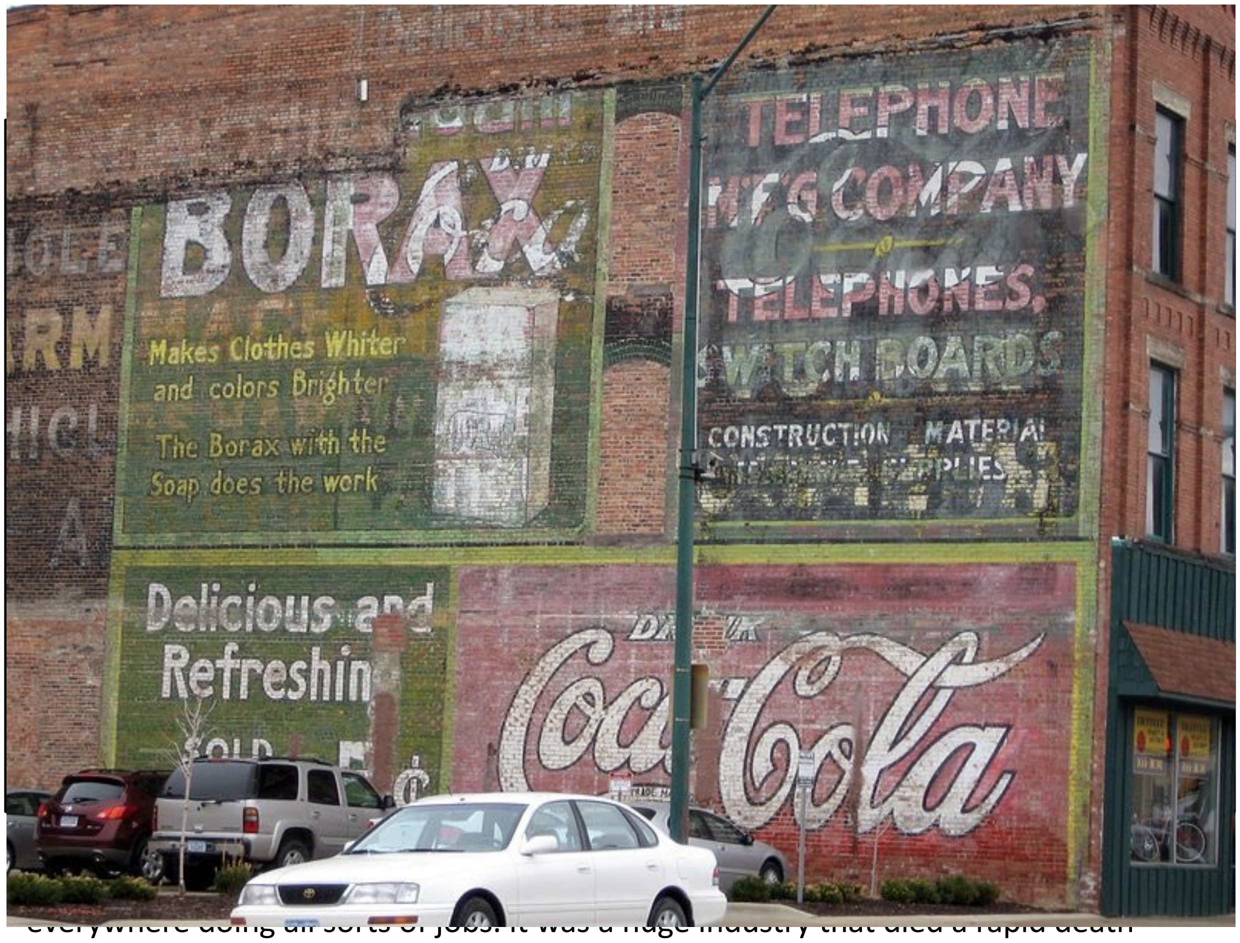
[grimace] yeesh

i would like to propose a theory of whether or not LLMs are a good thing that we can also apply to algorithms.

so for pretty much all of modern human history, if you wanted a sign, someone had to write it.

either you did it yourself or you paid someone to do it.

you'd put it over the door to your tavern, maybe, or you'd have someone paint on the inside of your window so they know what you sold inside.



everywhere doing all sorts of jobs. It was a huge industry that died a rapid death
thanks to this

And this

in the 90s and 2000s, suddenly you didn't need to pay someone to paint a sign for you. you could write it up yourself on a computer, print it off, and stick it up there yourself.

And so when was the last time you hired a sign painter?

sure, maybe these desktop printoffs didn't look as nice, but that didn't really matter.

they communicated the required information, and they did it basically for free. they weren't perfect, but they were _good enough._

GOOD ENOUGH

LLMs and [shudder] "AI" are the new desktop publishing platforms.

they're going to replace the things that we don't need something incredible for.

the "good enough" stuff. here's a concrete example — this image from my cover slide is AI-generated.

I wanted a picture, and the AI did a good enough job.

LLMs and [shudder] "AI" are the new desktop publishing platforms.

they're going to replace the things that we don't need something incredible for.

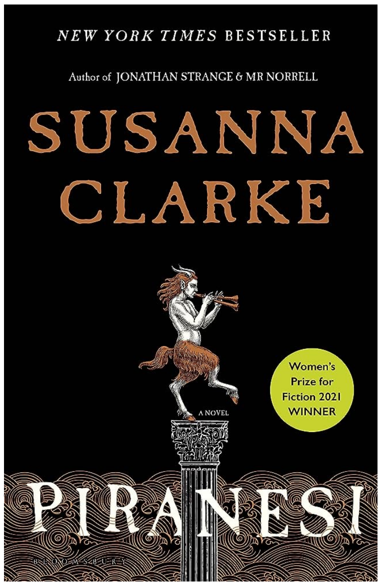
the "good enough" stuff. here's a concrete example — this image from my cover slide is AI-generated.

This one was too.

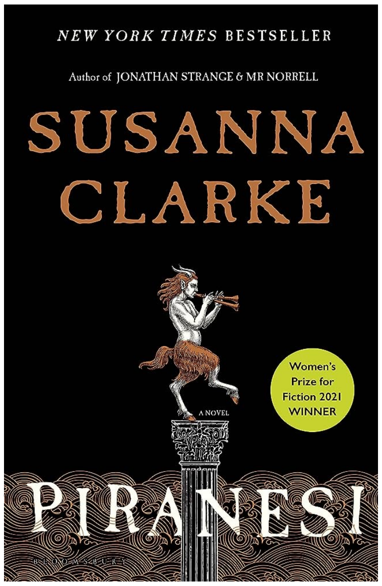
And this one too!

Here’s the thing about them — they’re just good enough. If I didn’t have a AI tool to make them, I just would have gone on google and found something similar.

If I want a painting that excites me, if I want a book that moves me, if i want a movie that thrills me


NEW YORK TIMES BESTSELLER
Author of JONATHAN STRANGE & MR NORRELL
SUSANNA CLARKE
A NOVEL
Women's Prize for Fiction 2021 WINNER
PIRANESI

If I want a painting that excites me, if I want a book that moves me, if i want a movie that thrills me


NEW YORK TIMES BESTSELLER
Author of JONATHAN STRANGE & MR NORRELL
SUSANNA CLARKE
A NOVEL
Women's Prize for Fiction 2021 WINNER
PIRANESI

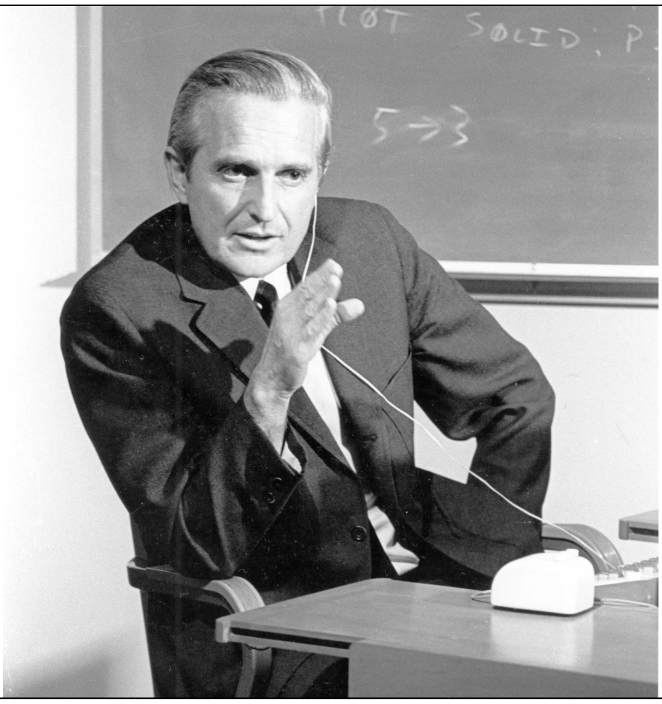
If I want a painting that excites me, if I want a book that moves me, if i want a movie that thrills me — I'm going to rely on a human for that.

But if I need to bang out a boilerplate email to someone and an LLM can do that? hell yeah, that's good enough for me.

it saves me time. it saves me effort. it becomes an _enhancement_ for me.

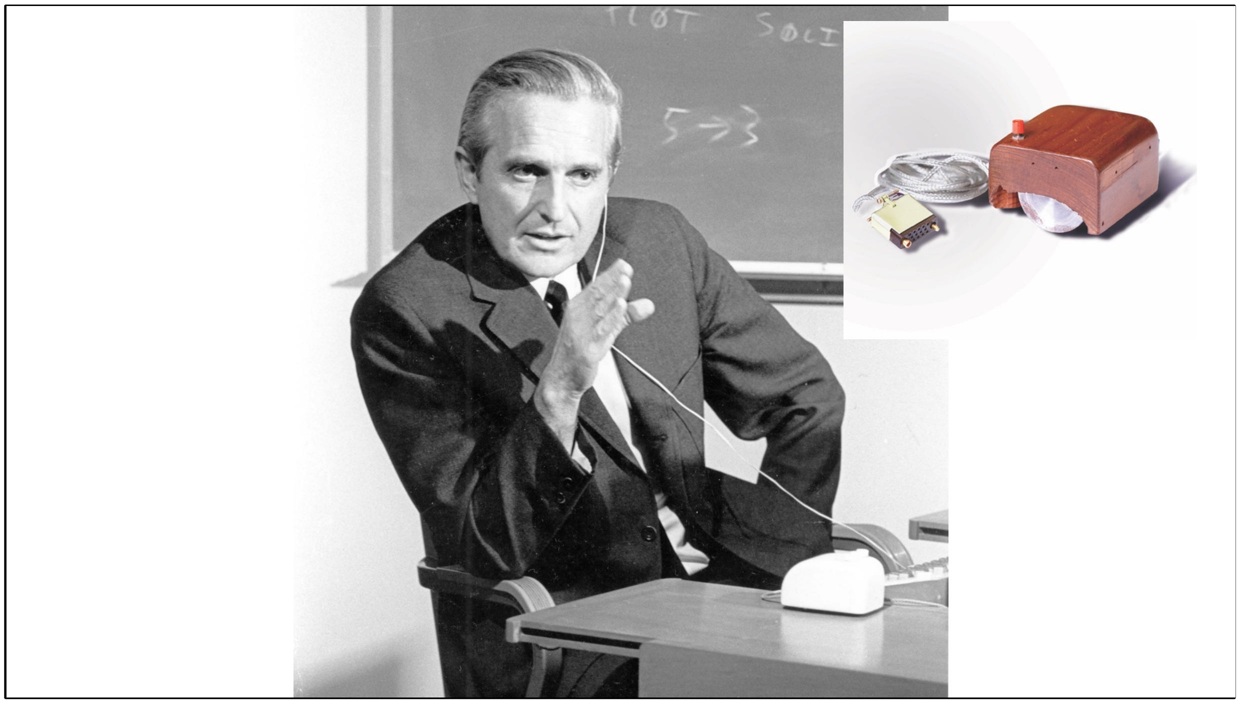
it's like any other good tool: it becomes a part of how i think about and solve problems.

and that's what your algorithms should be doing too. They should be enhancing your users, not replacing them.

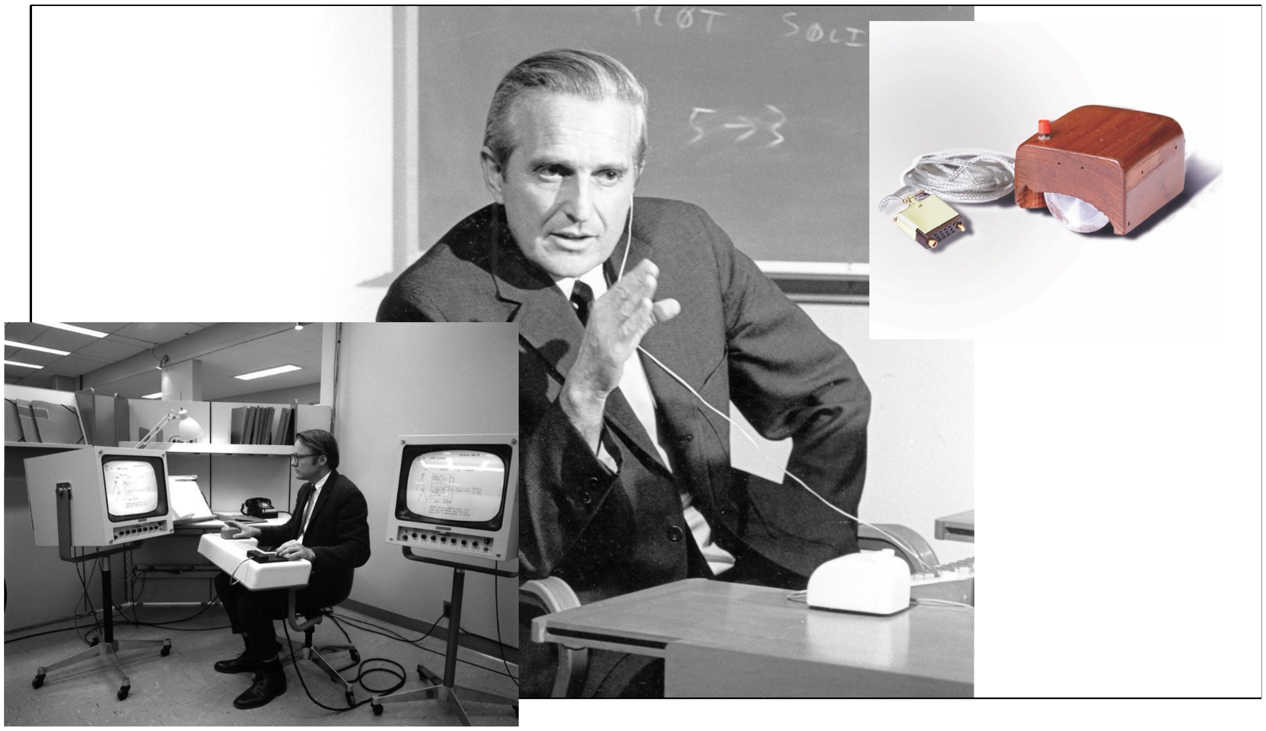


this idea comes from doug engelbart, an absolute titan in the field of human-computer interaction.

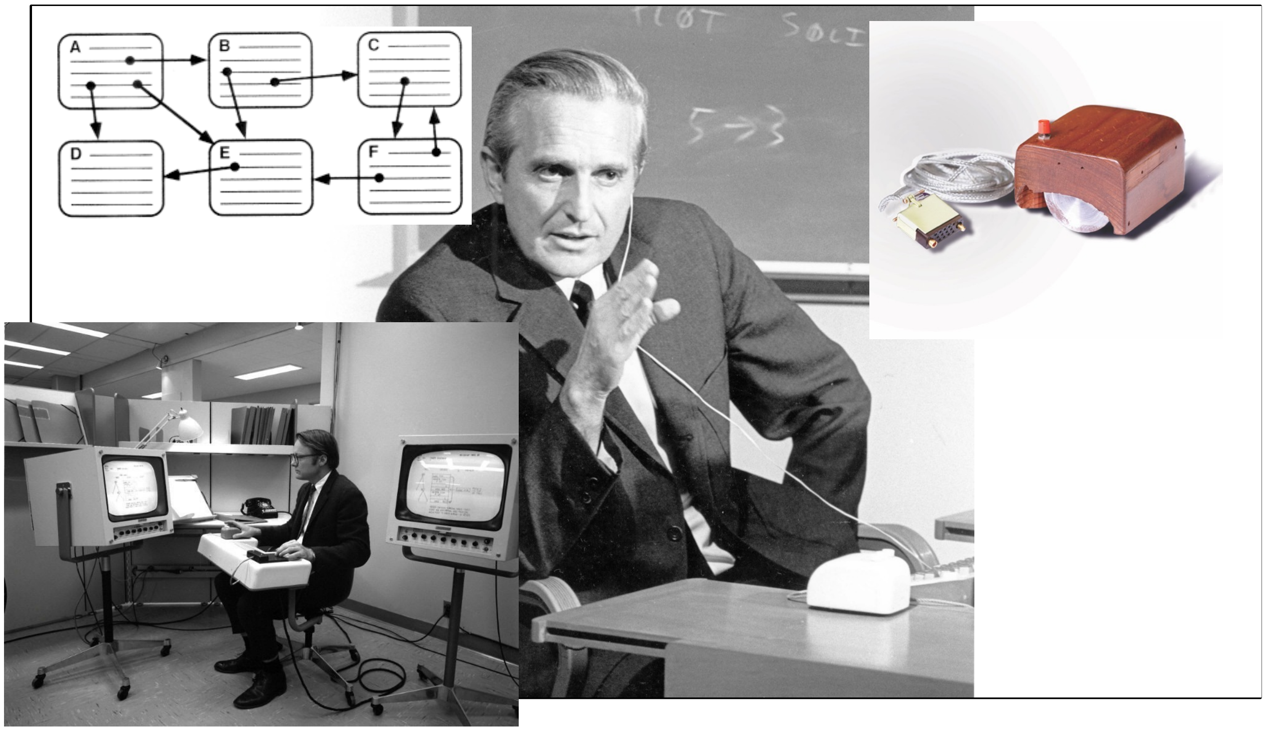
he invented the mouse,



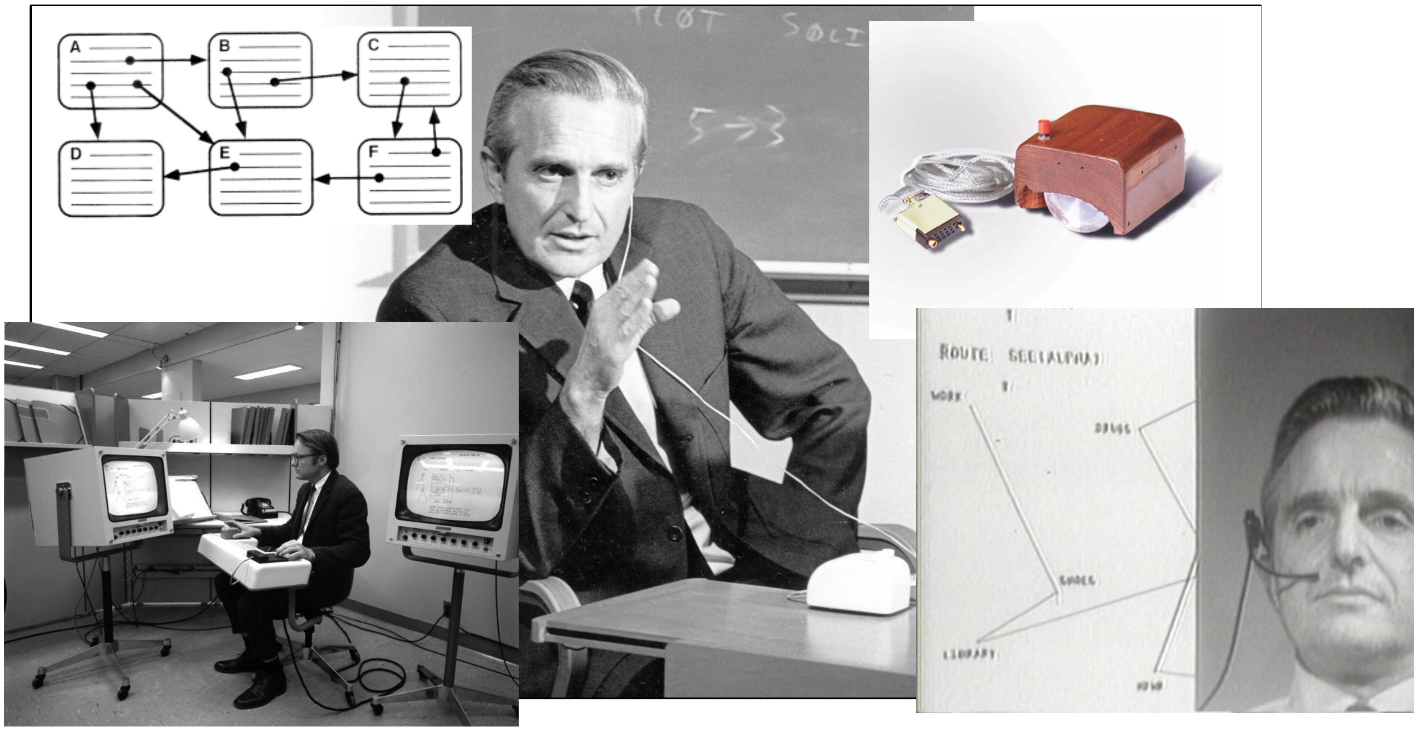
this idea comes from doug engelbart, an absolute titan in the field of human-computer interaction. he invented the mouse, the word processor,



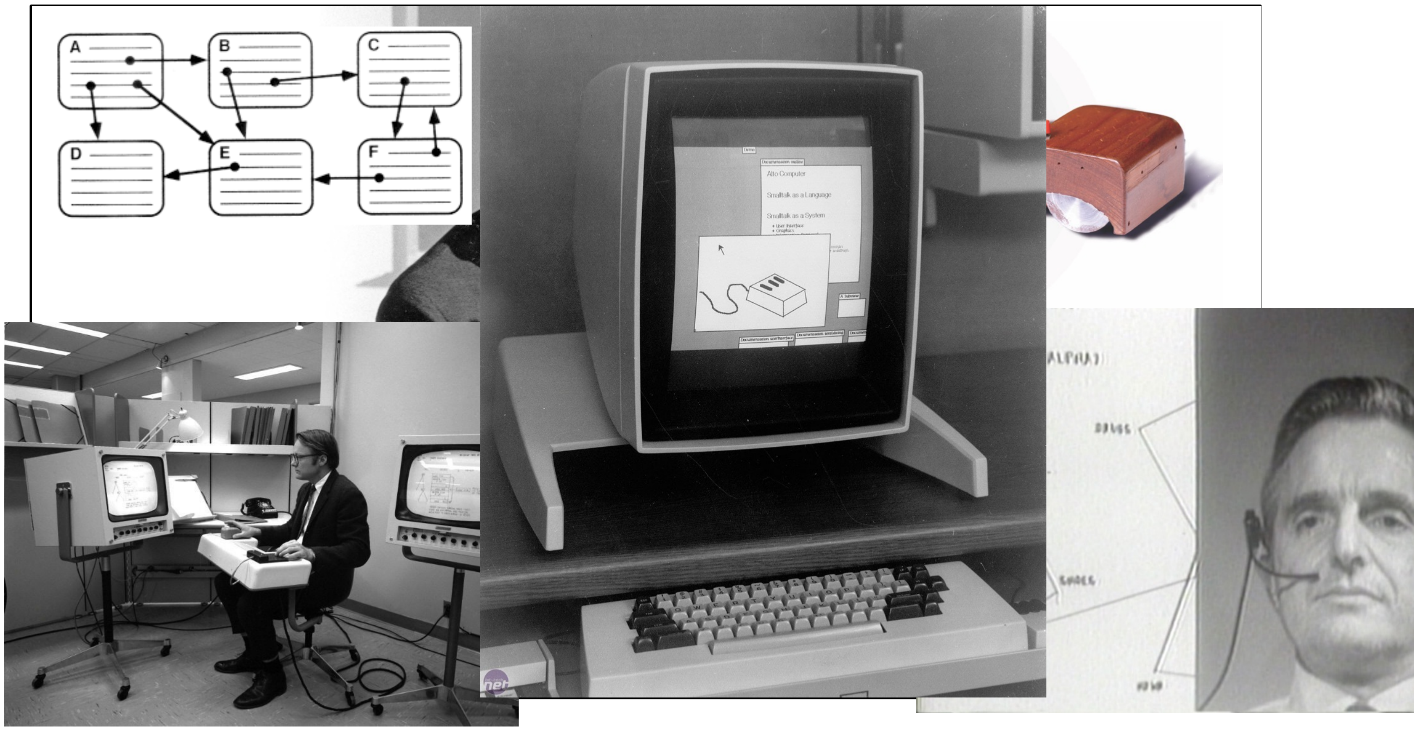
this idea comes from doug engelbart, an absolute titan in the field of human-computer interaction. he invented the mouse, the word processor, hypertext



this idea comes from doug engelbart, an absolute titan in the field of human-computer interaction. he invented the mouse, the word processor, hypertext, networked computers — aka the internet —



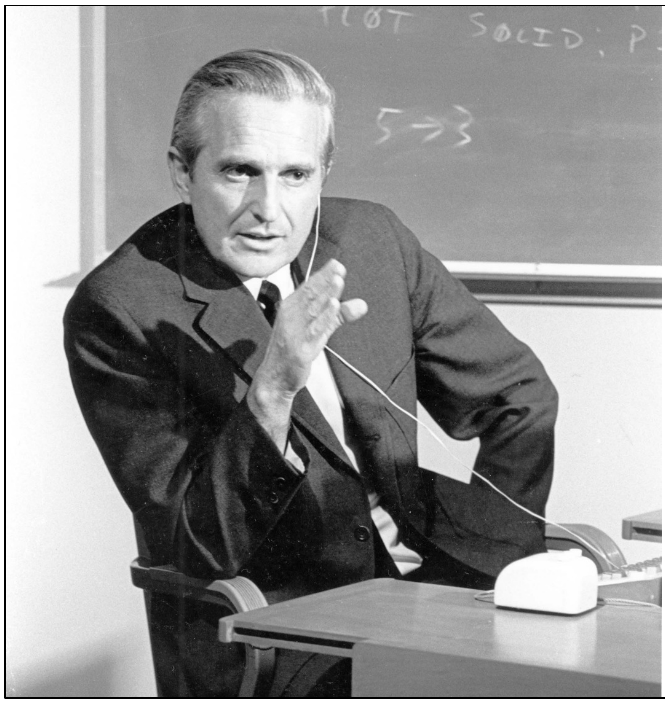
this idea comes from doug engelbart, an absolute titan in the field of human-computer interaction. he invented the mouse, the word processor, hypertext, networked computers — aka the internet — the graphical user interface



— and he did it basically all in one presentation. now it's called the "mother of all demos."

go look it up — it's one of those things that's almost not impressive anymore because everything he shows is so common now.

but he was doing it sixty years ago.



"The better we get at getting better, the faster we will get better."

–Doug Engelbart

so he had this idea of augmented intelligence.

if we think of tools as augments to what we can already do — a hammer lets me swing with more force and precision than my hand allows —

then a computer should be a tool that augments our intelligence.

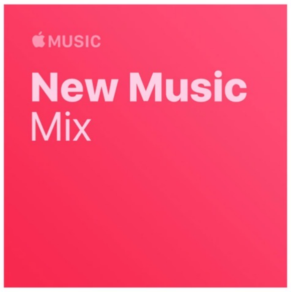
we should be able to think better, filter more data, make more informed decisions —

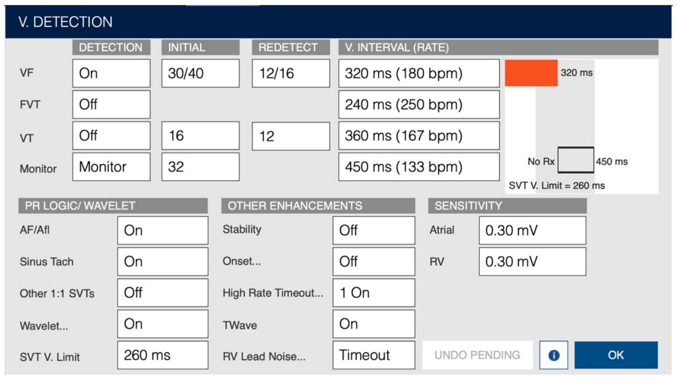
the computer should be supporting that by doing all the boring scut work we normally have to do.

BORING

GOOD ENOUGH

if it can do a GOOD ENOUGH job on its own, we can replace a lot of the boring stuff with an algorithm.





V. DETECTION

	DETECTION	INITIAL	REDETECT	V. INTERVAL (RATE)
VF	On	30/40	12/16	320 ms (180 bpm)
FVT	Off			240 ms (250 bpm)
VT	Off	16	12	360 ms (167 bpm)
Monitor	Monitor	32		450 ms (133 bpm)

PR LOGIC/ WAVELET		OTHER ENHANCEMENTS		SENSITIVITY	
AF/Afl	On	Stability	Off	Atrial	0.30 mV
Sinus Tach	On	Onset...	Off	RV	0.30 mV
Other 1:1 SVTs	Off	High Rate Timeout...	1 On		
Wavelet...	On	TWave	On		
SVT V. Limit	260 ms	RV Lead Noise...	Timeout		

That’s how you build an algorithm. To save your users time and energy.

now they don't have to learn linear algebra to get infinite music recommendations.

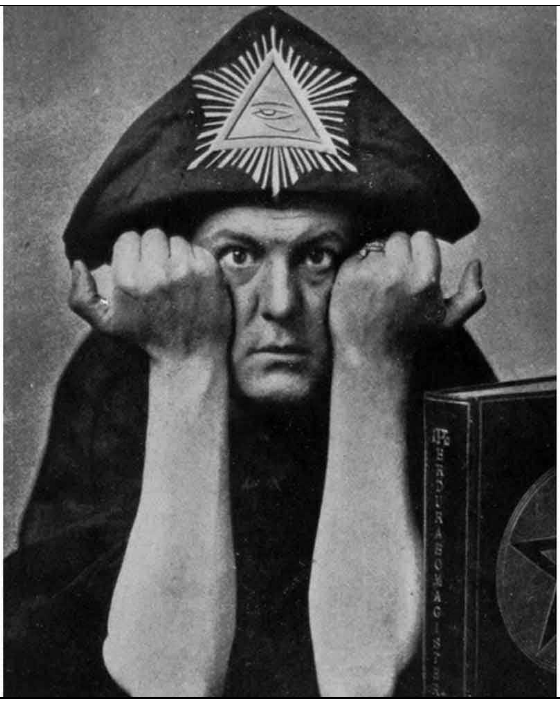
now they don't have to be experts on cardiac electrophysiology to stop someone from dying.

you've taken a hard part of their job and made it easier, and in doing so you've freed up their time to do more complex things, more interesting things, the same jobs but better.

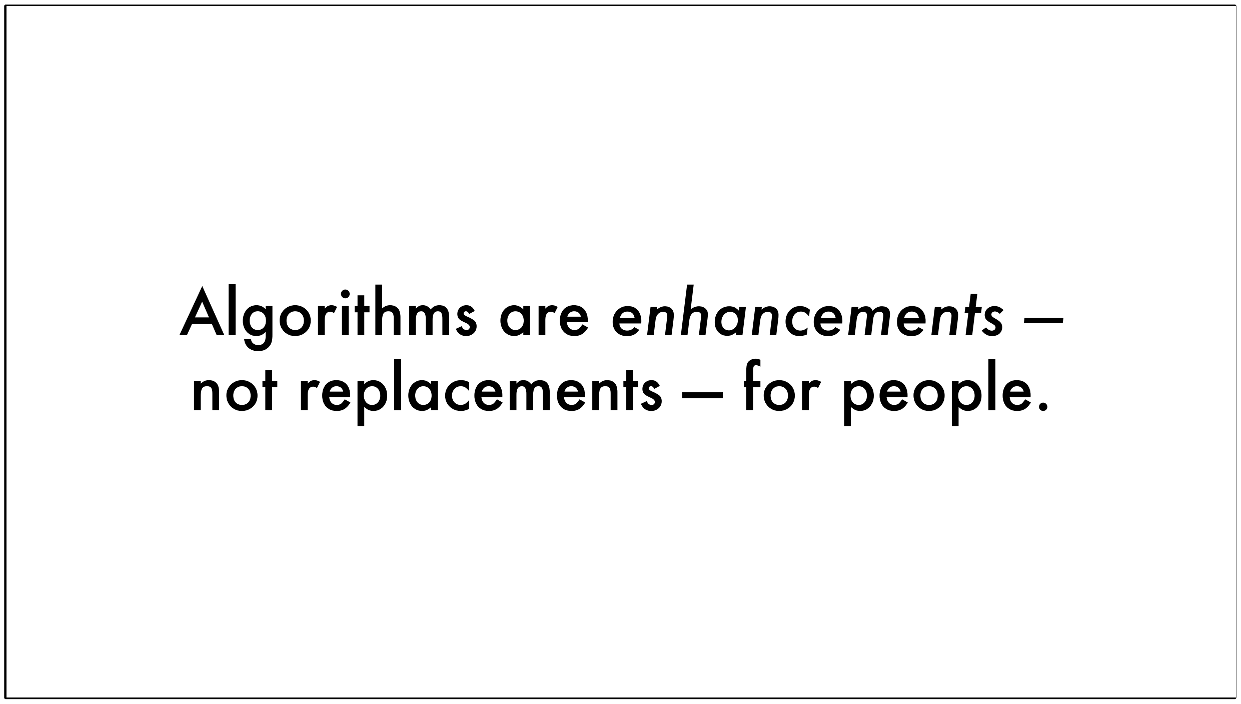
and this is how you explain it. This is how you *build* it

algorithms aren't coming for your paycheck, they're coming for your podcasts.

now all the time you used to spend listening to them while you did mindless things for your job are replaced by a computer and you get to focus on the interesting stuff.



if there's one other thing i want you to remember from this talk besides crowley,

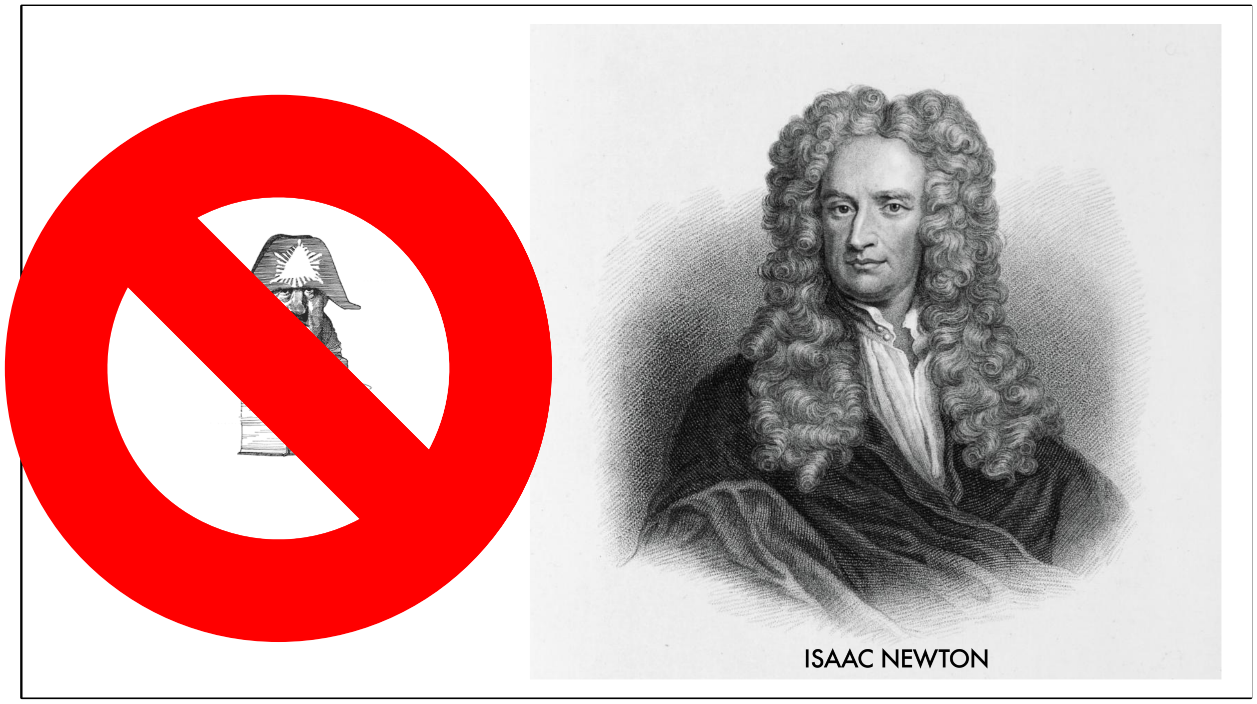


Algorithms are *enhancements* –
not replacements – for people.

it's this:

an algorithm is an *enhancement,* not a _replacement,_ for a person.

And you should be building your interfaces with that in mind.



we make this palatable to users through developing trust, not faith.

don't preach about how this algorithm is going to save them time, _show them._

show them every time they interact with the system by revealing the right things to them,

giving them the right level of control,

by letting them see how A turns into Z.

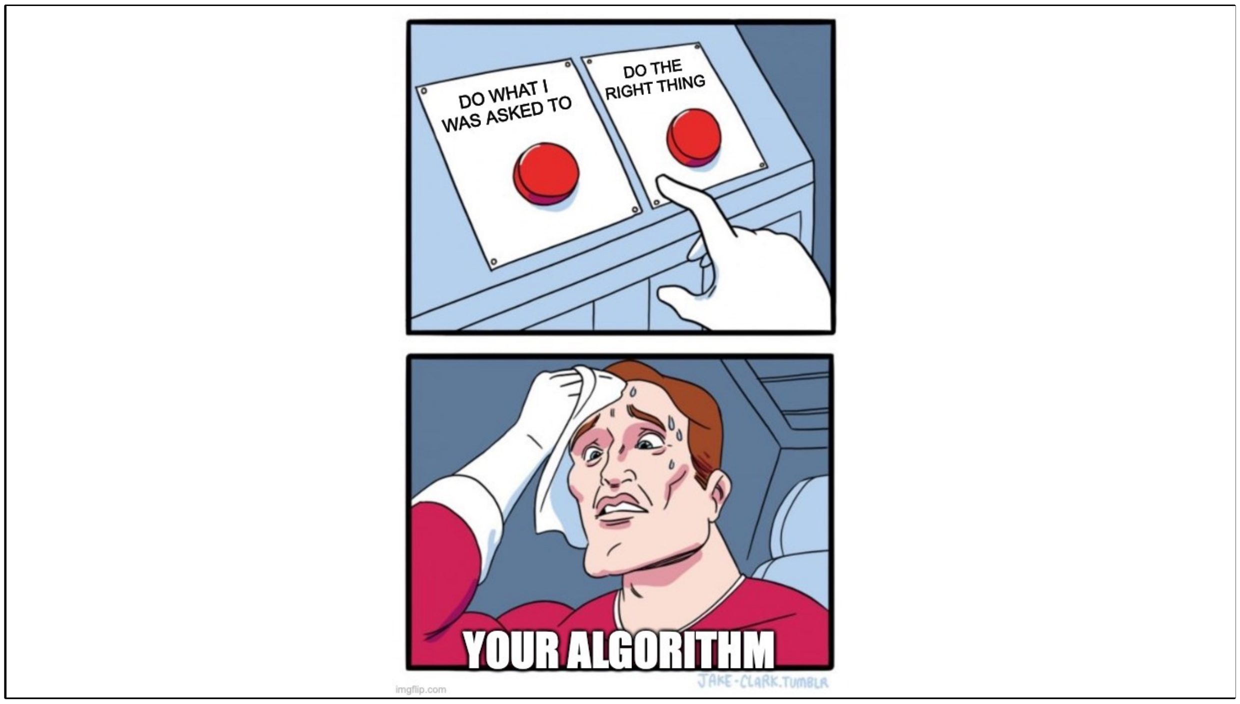
don't make them crack open the Book of Thoth every time they want answers for how the system works.

think about how angry alastair is going to be when the answers are already provided for them.

i hope i've demonstrated why trust ential.

because sometimes the algorithm is going to give you an output you didn't expect, and it's going to be _right._

and in that moment, your user is going to have to trust it, because it might be a life-or-death decision.

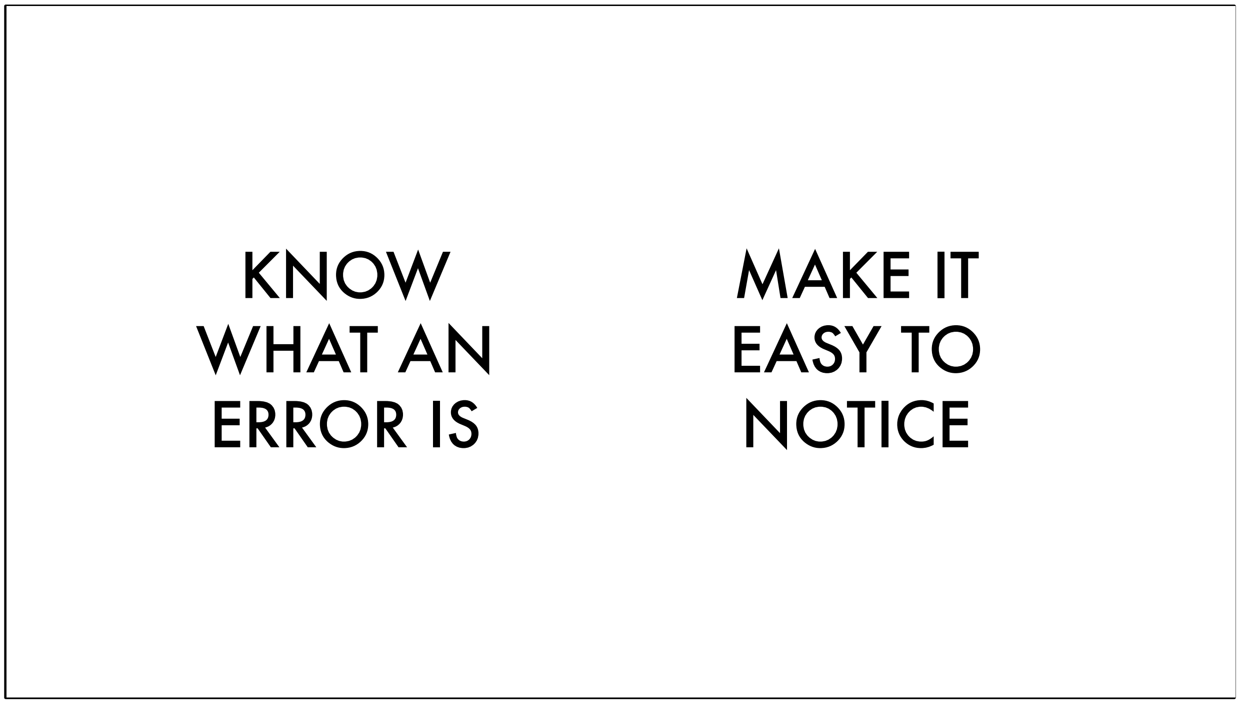


despite our best abilities, things will go wrong.

an algorithm will make a bad call.

someone will enter bad data and get bad data back.

your job as a designer is twofold: it’s to identify what the errors are and make them easy for your usrs to idetnfiy

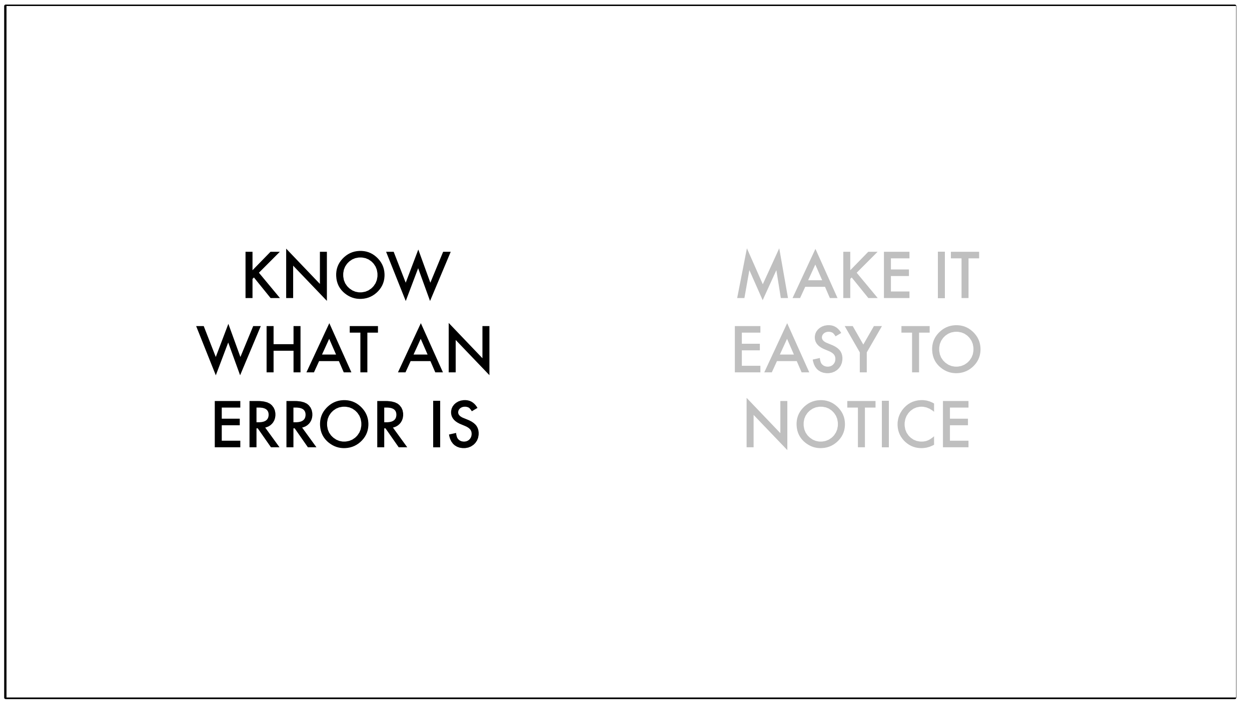


KNOW WHAT AN ERROR IS

MAKE IT EASY TO NOTICE

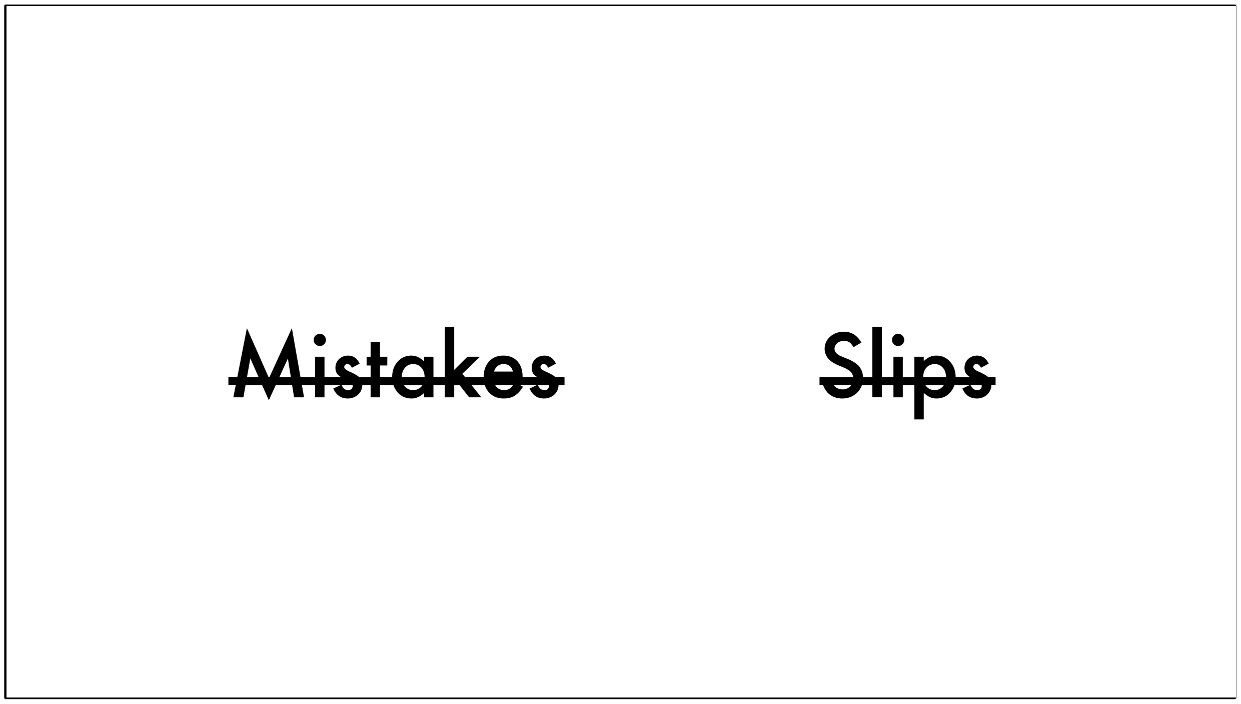
it's to minimize the likelihood bad things happen, and to make them easy to fix when they do.

with an algorithm, this can be surprisingly challenging.


KNOW WHAT AN ERROR IS

MAKE IT EASY TO NOTICE

the first thing you have to do is come up with a definition: what even counts as an _error_ here?

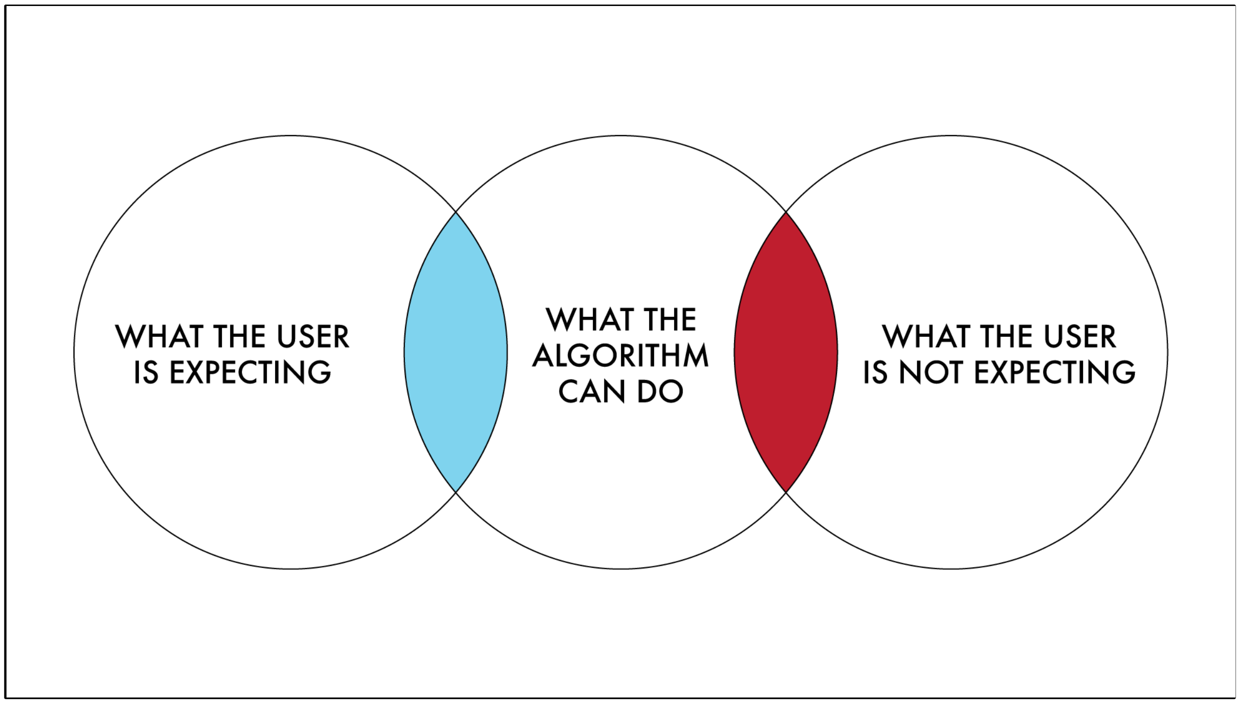


~~Mistakes~~ ~~Slips~~

we have to remember that we're not defining human errors here.

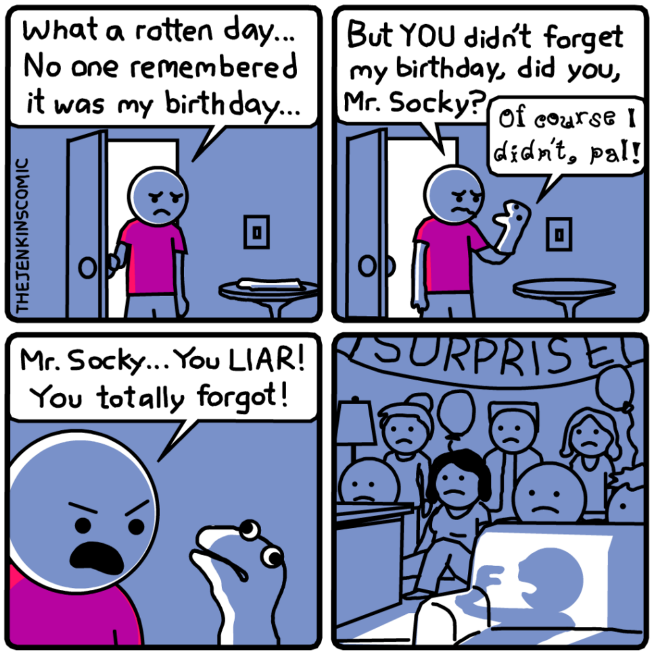
algorithms don't make mistakes and they don't have slips.

an error in this case is where there is a difference between


WHAT THE USER IS EXPECTING
WHAT THE ALGORITHM CAN DO
WHAT THE USER IS NOT EXPECTING

what the user is _expecting_ the algorithm to provide and what the system actually provided.

And let’s be really clear about this for a second — there is a difference between an *error* and an unexpected result.


What a rotten day... No one remembered it was my birthday...
THEJENKINSCOMIC
But YOU didn't forget my birthday, did you, Mr. Socky?
Of course I didn't, pal!
Mr. Socky... You LIAR! You totally forgot!
SURPRISE

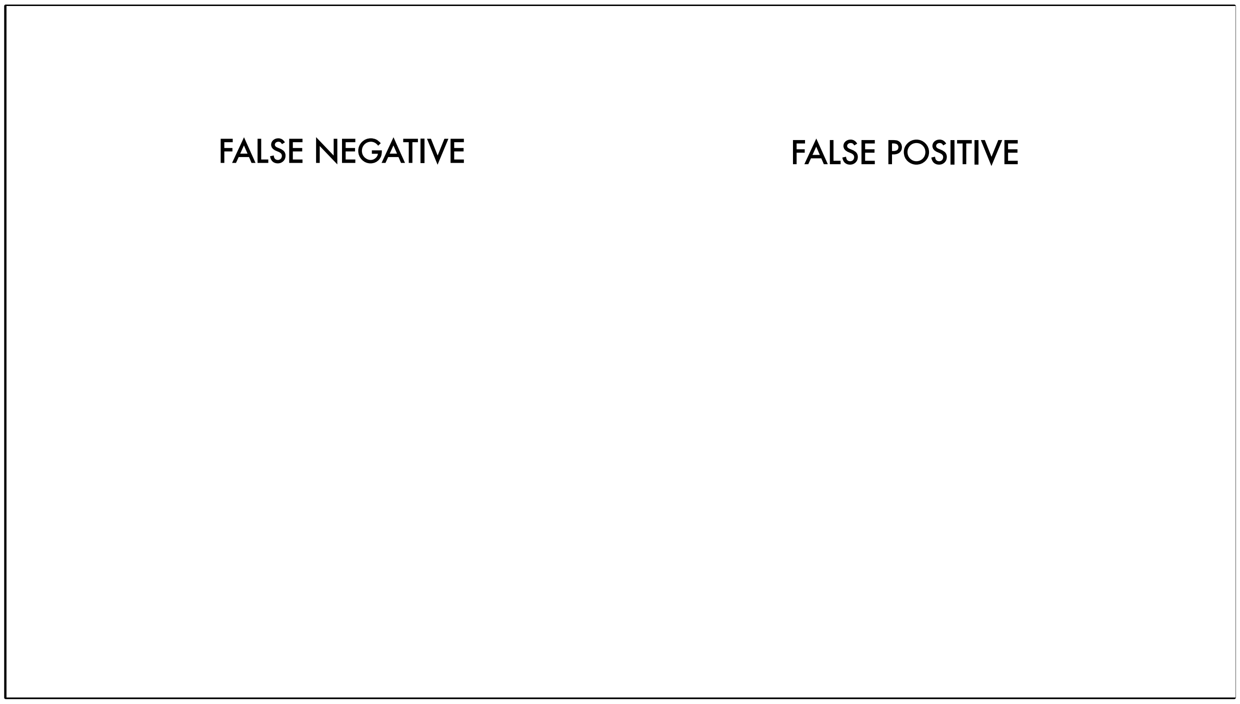
Unexpected results are surprises!

[Wait 7 seconds]

Surprising Results ≠ Errors

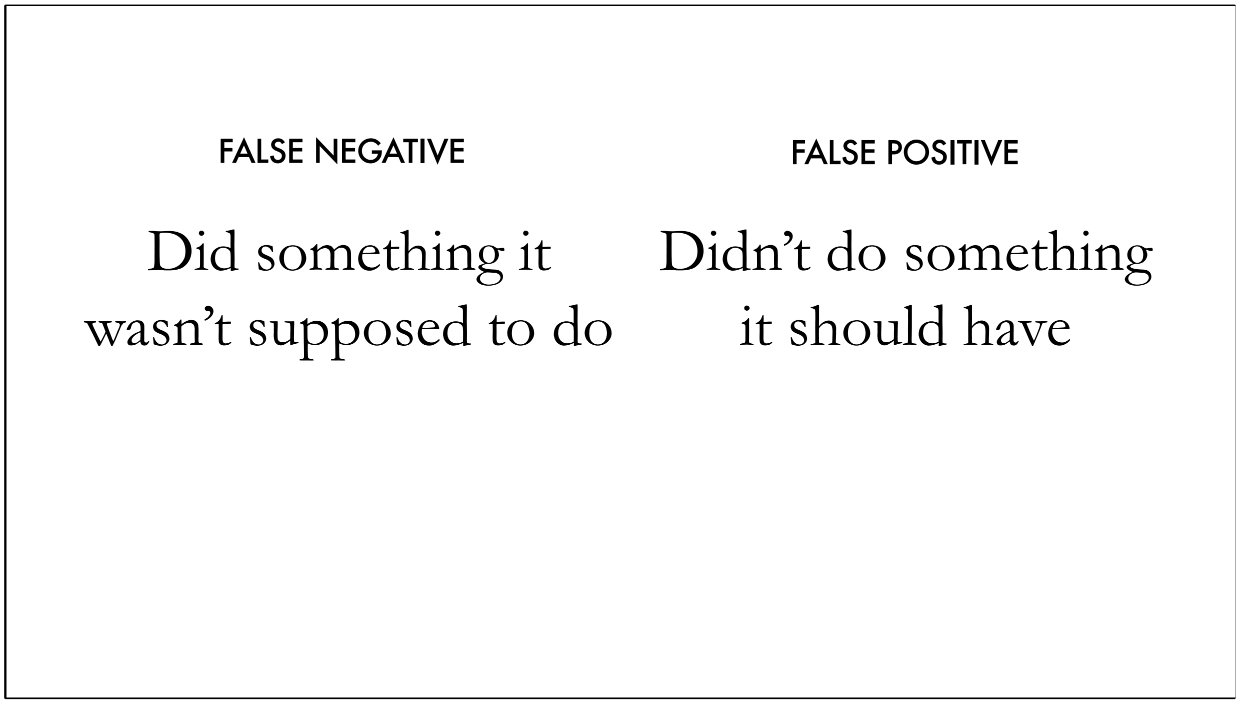
But just because the algorithm did something you didn’t expect doesn’t mean it was *wrong*.

You have to define what counts as incorrect. And usually, algorithmic errors fall into two categories



FALSE NEGATIVE FALSE POSITIVE

false negatives and _false positives_. there are certainly more errors that can happen, but this is a nice framework generally.



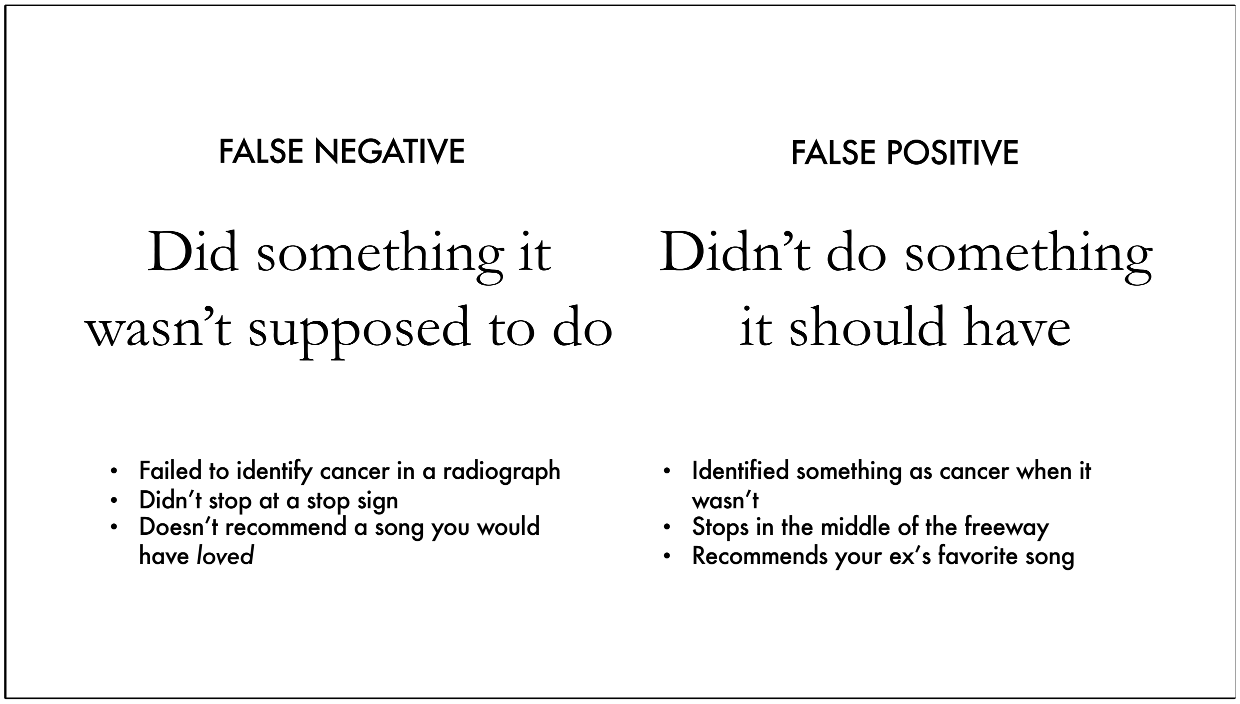
FALSE NEGATIVE

Did something it wasn’t supposed to do

FALSE POSITIVE

Didn’t do something it should have

a false negative is when the system doesn't do something it was supposed to, and a false positive is when it does something it shouldn't have.



FALSE NEGATIVE

Did something it wasn't supposed to do

- Failed to identify cancer in a radiograph
- Didn't stop at a stop sign
- Doesn't recommend a song you would have *loved*

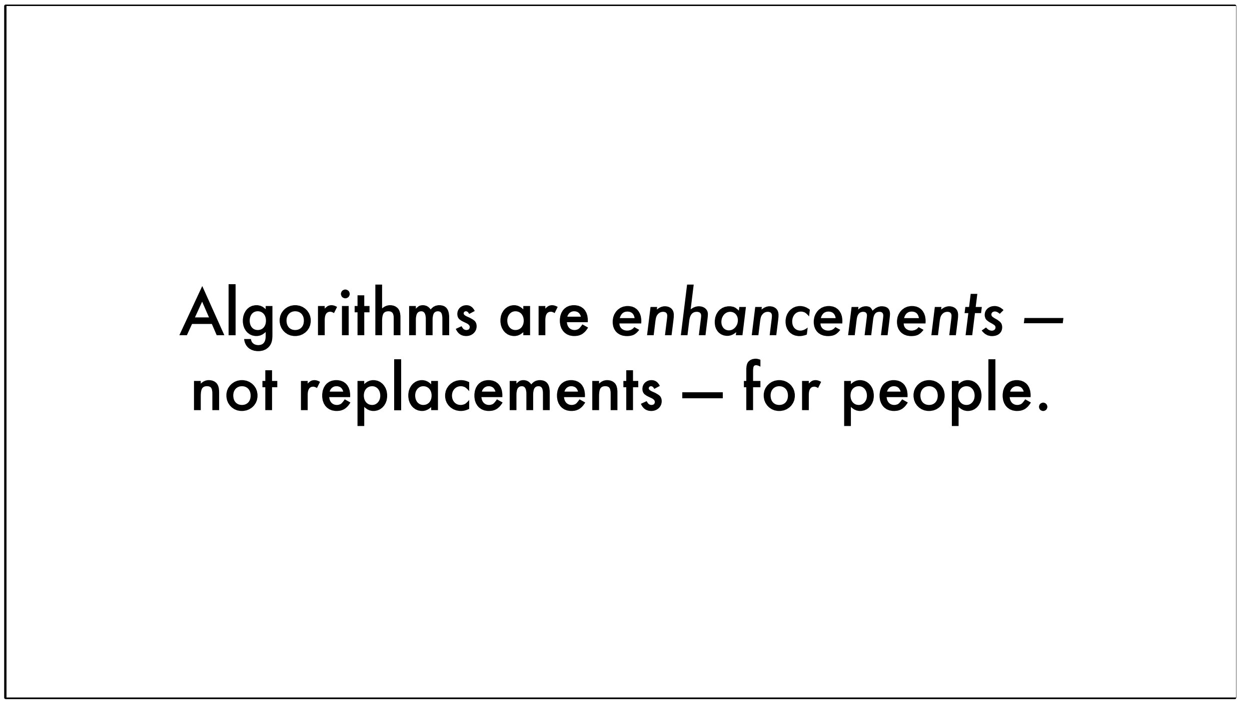
FALSE POSITIVE

Didn't do something it should have

- Identified something as cancer when it wasn't
- Stops in the middle of the freeway
- Recommends your ex's favorite song

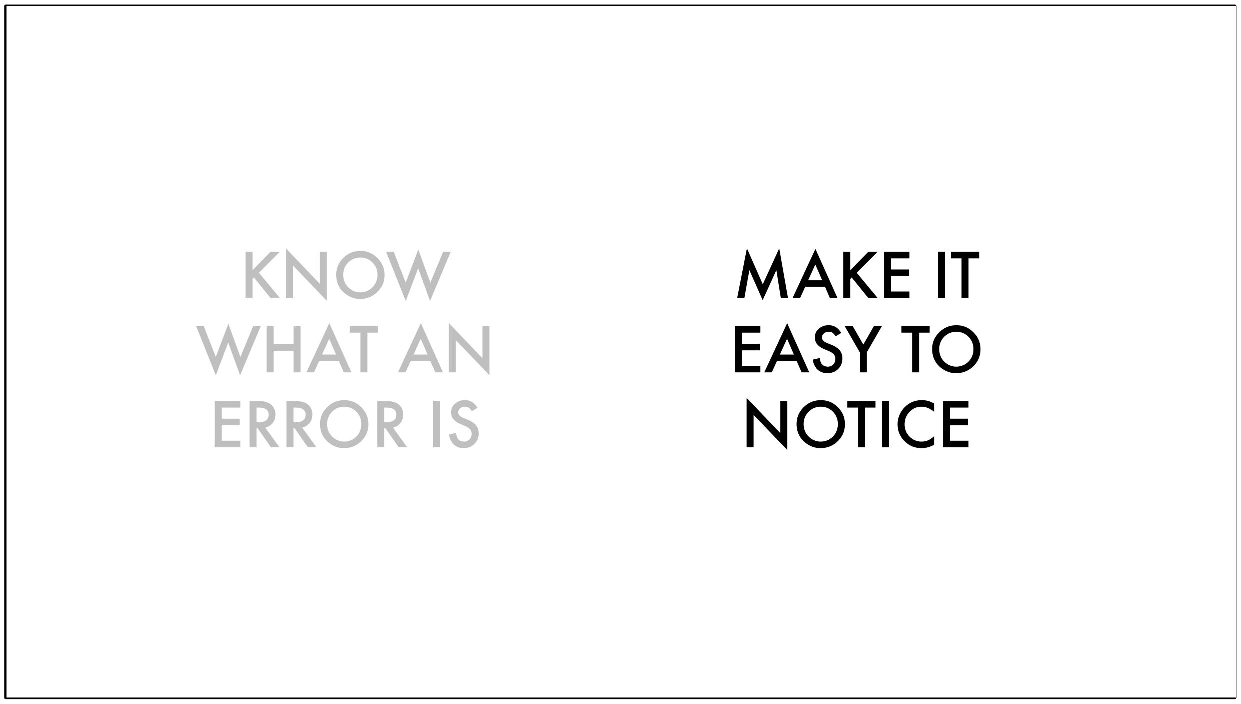
and the really tricky part here is that by definition, the system can't tell that an error happened — if it did, then it could have prevented it.

the only way that an error can be identified is if a human notices — and if the data is available.



Algorithms are *enhancements* – not replacements – for people.

which is why algorithms are tools that ENHANCE, not replace, and we build systems we can UNDERSTAND.

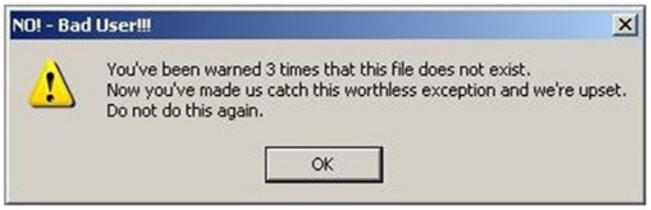


KNOW WHAT AN ERROR IS

MAKE IT EASY TO NOTICE

So you got your surprising result. Now it needs to be checked to make sure an error didn't occur.

Someone has to sit down and diagnose the output.



You've been warned 3 times that this file does not exist.
Now you've made us catch this worthless exception and we're upset.
Do not do this again.

The first thing you have to do is alert the user that an error even happened.

This is not always easy — sometimes an algorithm knows when it didn't work.

Maybe the input data was bad or outside a range it could work with.

If the algorithm can tell something went wrong, tell the user and give them some advice.

This is basic error handling stuff I won't get into here.

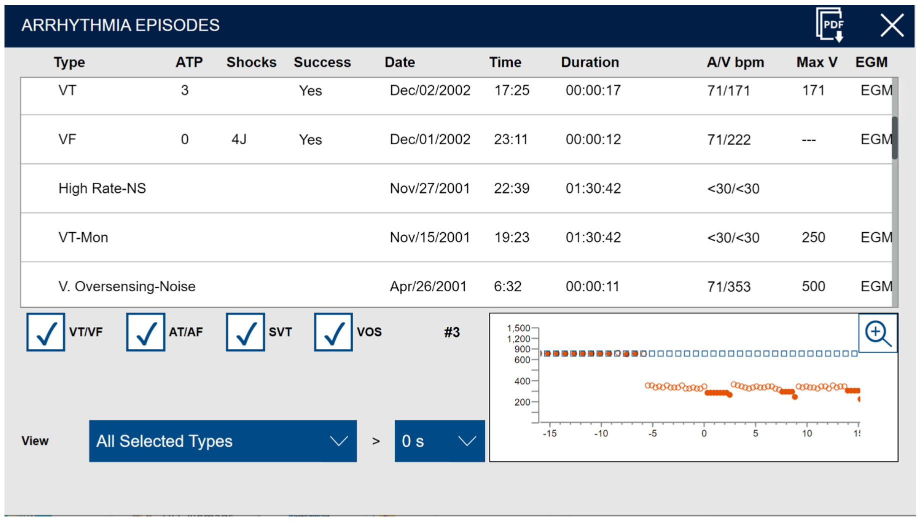
What's much trickier is when the system made a mistake and it can't tell.

The only way to know something went wrong is to provide a backup to the user — and have someone looking for it.

You have to present _all_ the outputs of an algorithm to the user,

which usually means they need some tools to sort through the outputs to get what they're looking for.

Most of the time the algorithm does this for you — but when something unexpected happens, you need to get in there yourself and take a look


ARRHYTHMIA EPISODES

Type	ATP	Shocks	Success	Date	Time	Duration	A/V bpm	Max V	EGM
VT	3		Yes	Dec/02/2002	17:25	00:00:17	71/171	171	EGM
VF	0	4J	Yes	Dec/01/2002	23:11	00:00:12	71/222	---	EGM
High Rate-NS				Nov/27/2001	22:39	01:30:42	<30/<30		
VT-Mon				Nov/15/2001	19:23	01:30:42	<30/<30	250	EGM
V. Oversensing-Noise				Apr/26/2001	6:32	00:00:11	71/353	500	EGM

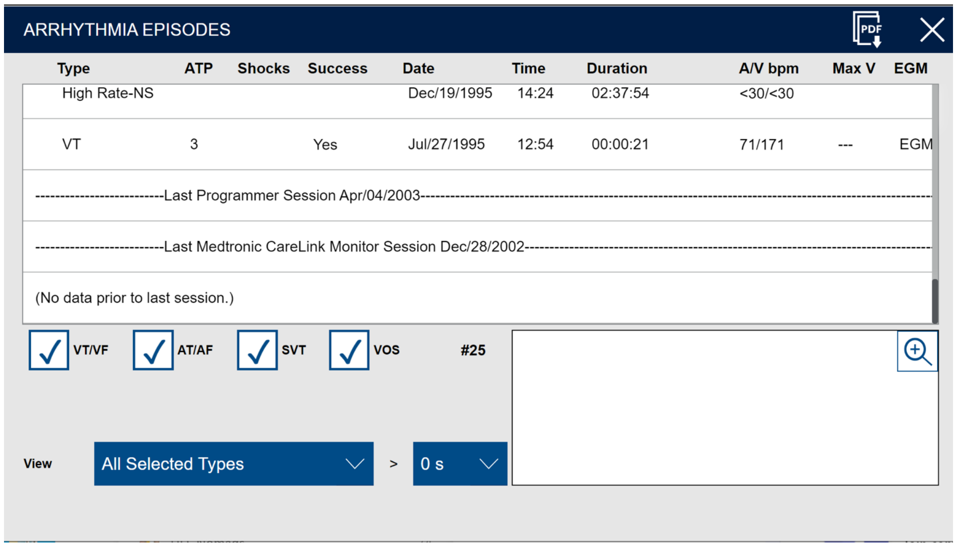
✓ VT/VF ✓ AT/AF ✓ SVT ✓ VOS #3

View: All Selected Types > 0 s

back in heart-world, we keep track of every time the device detects what it thinks is a dangerous heartbeat.

it stores the information about what the heart did, how it classified it, and what happened during the event.

this information is all available, and the user has ways of searching through it to get what they need.



ARRHYTHMIA EPISODES

Type	ATP	Shocks	Success	Date	Time	Duration	A/V bpm	Max V	EGM
High Rate-NS				Dec/19/1995	14:24	02:37:54	<30/<30		
VT	3		Yes	Jul/27/1995	12:54	00:00:21	71/171	---	EGM

-------------------------Last Programmer Session Apr/04/2003-------------------------

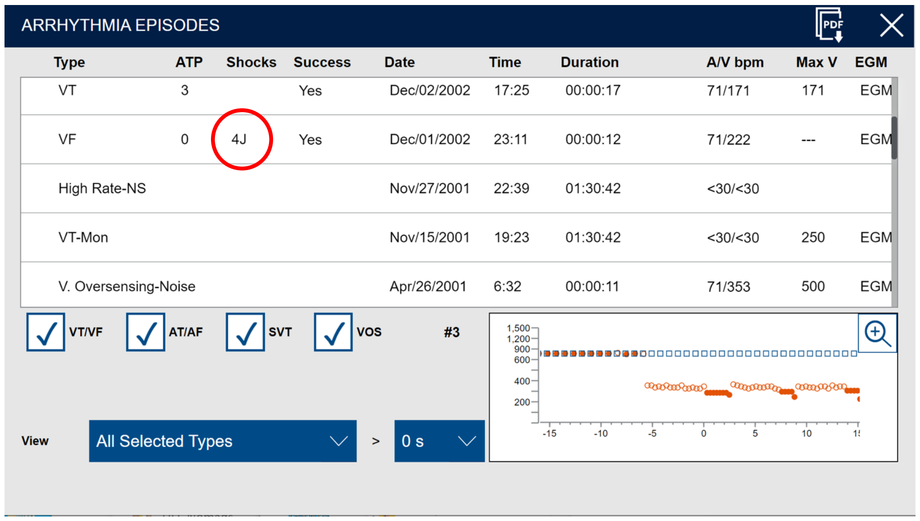
-------------------------Last Medtronic CareLink Monitor Session Dec/28/2002-------------------------

(No data prior to last session.)

✓ VT/VF ✓ AT/AF ✓ SVT ✓ VOS #25

View: All Selected Types > 0 s

in this case, our users are curious about what happened since they last saw this patient,



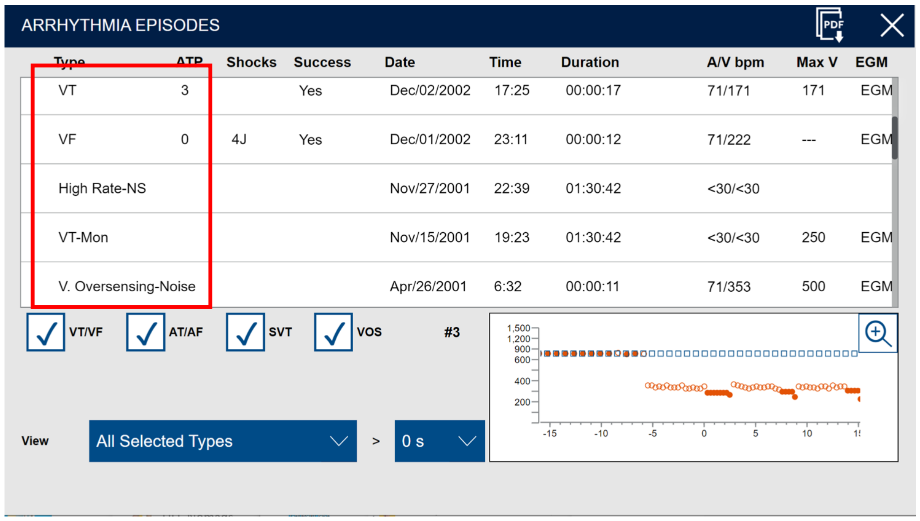
ARRHYTHMIA EPISODES

Type	ATP	Shocks	Success	Date	Time	Duration	A/V bpm	Max V	EGM
VT	3		Yes	Dec/02/2002	17:25	00:00:17	71/171	171	EGM
VF	0	4J	Yes	Dec/01/2002	23:11	00:00:12	71/222	---	EGM
High Rate-NS				Nov/27/2001	22:39	01:30:42	<30/<30		
VT-Mon				Nov/15/2001	19:23	01:30:42	<30/<30	250	EGM
V. Oversensing-Noise				Apr/26/2001	6:32	00:00:11	71/353	500	EGM

✓ VT/VF ✓ AT/AF ✓ SVT ✓ VOS #3

View: All Selected Types > 0 s

what arrhythmias had shocks,



ARRHYTHMIA EPISODES

Type	ATP	Shocks	Success	Date	Time	Duration	A/V bpm	Max V	EGM
VT	3		Yes	Dec/02/2002	17:25	00:00:17	71/171	171	EGM
VF	0	4J	Yes	Dec/01/2002	23:11	00:00:12	71/222	---	EGM
High Rate-NS				Nov/27/2001	22:39	01:30:42	<30/<30		
VT-Mon				Nov/15/2001	19:23	01:30:42	<30/<30	250	EGM
V. Oversensing-Noise				Apr/26/2001	6:32	00:00:11	71/353	500	EGM

✓ VT/VF ✓ AT/AF ✓ SVT ✓ VOS #3

View: All Selected Types > 0 s

and the specific kinds of rhythms that this patient is likely to have.

the rest is important, so we still show it, but we don't put it front and center.

this UI is built around giving the user the tools they need to understand the algorithm.

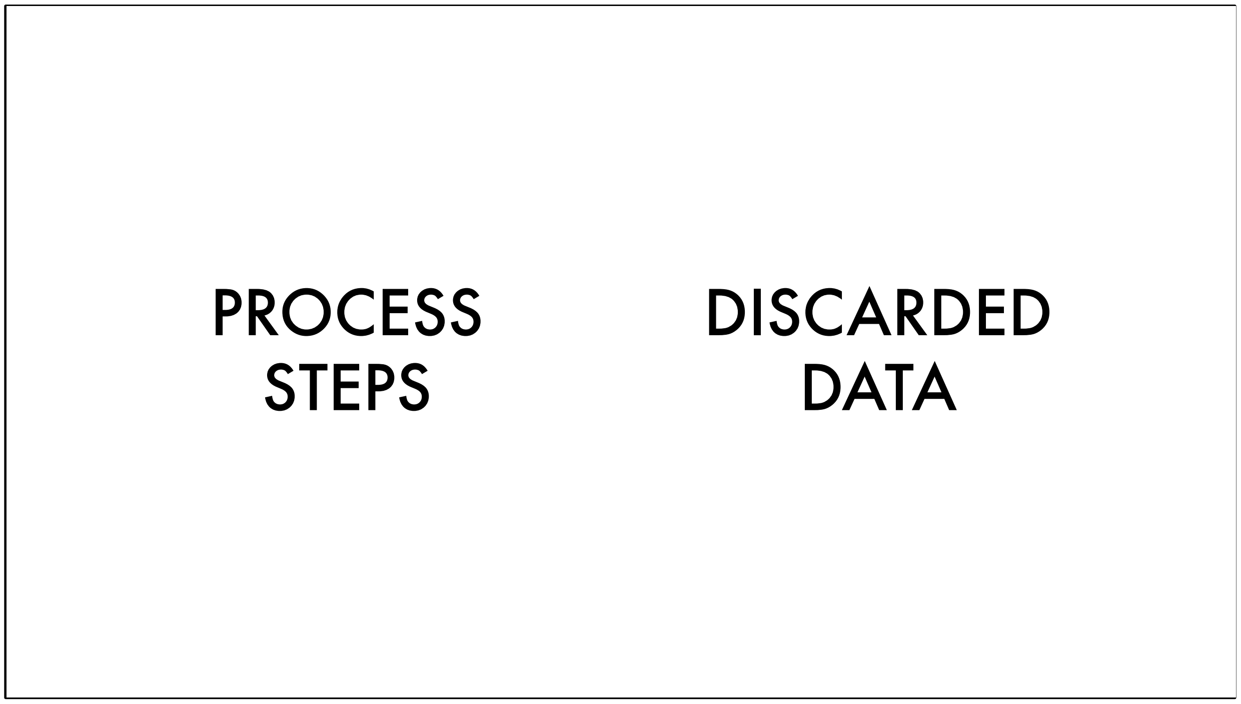
It can be really tempting to just discard the things we don't really care about.

If the point of an algorithm is to save the user's time, we shouldn't give them the option to examine what we discarded, right?

That's a very aleister way of thinking.

If you have a 1% failure rate, 99% of the time your users never have to dig through the trash.

But in that 1% of cases, it's vitally important so users can understand what went wrong.



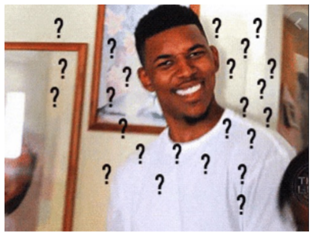
PROCESS STEPS

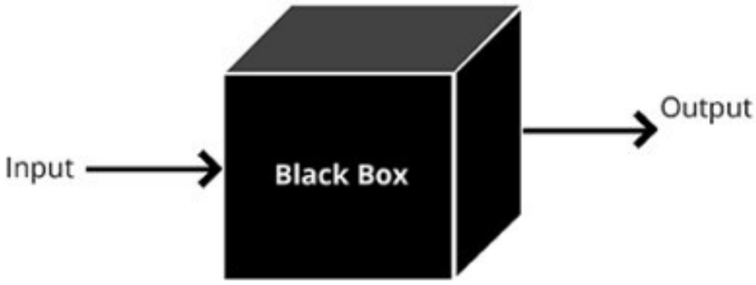
DISCARDED DATA

Usually what you need to show them is

How the algorithm reached its decision and what it did with what it analyzed

This allows your users to check that everything worked the way it was supposed to and confirm that there's nothing inappropriately ignored.





Remember, you don’t want to be a black box.

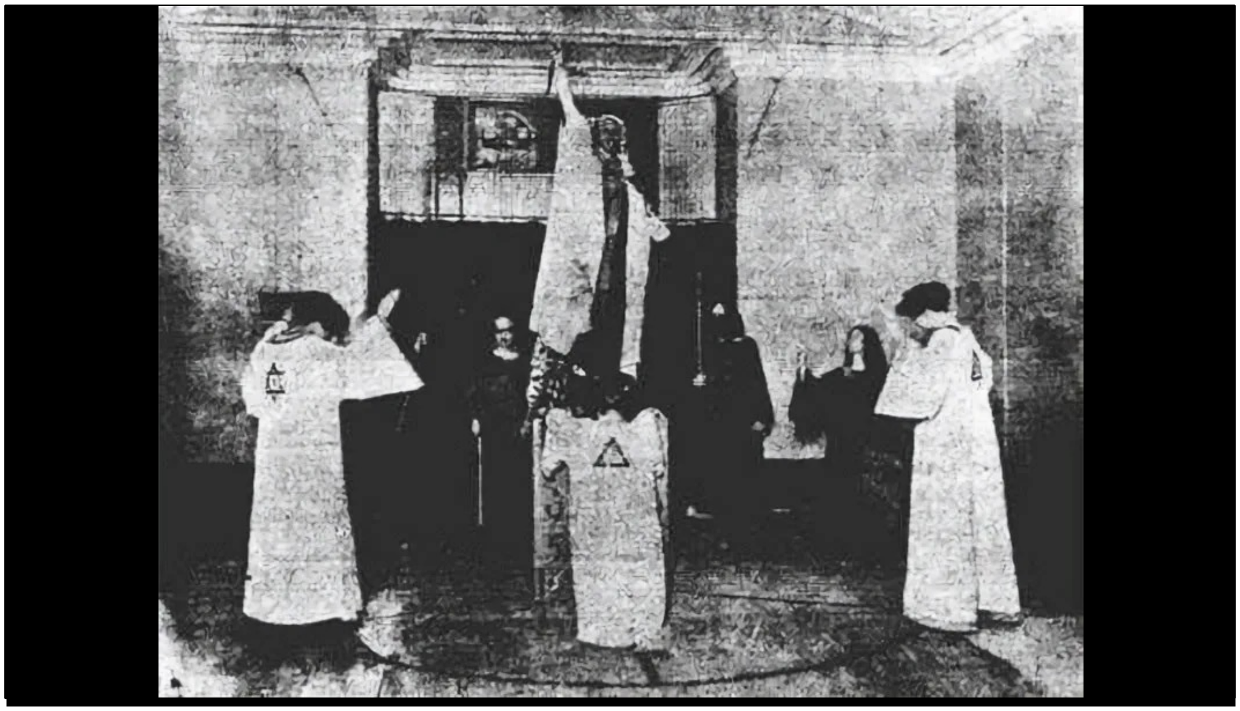
so a huge amount of building trust is not getting rid of stuff and making it easy for your users to understand what's going on.

it's important that users are aware of not only when things work, but also when things go wrong.

it's important when building an answer space to be aware of what the wrong answers are just as much as you know about the right ones.

Black boxes lead to faith, not trust.

Black boxes lead to



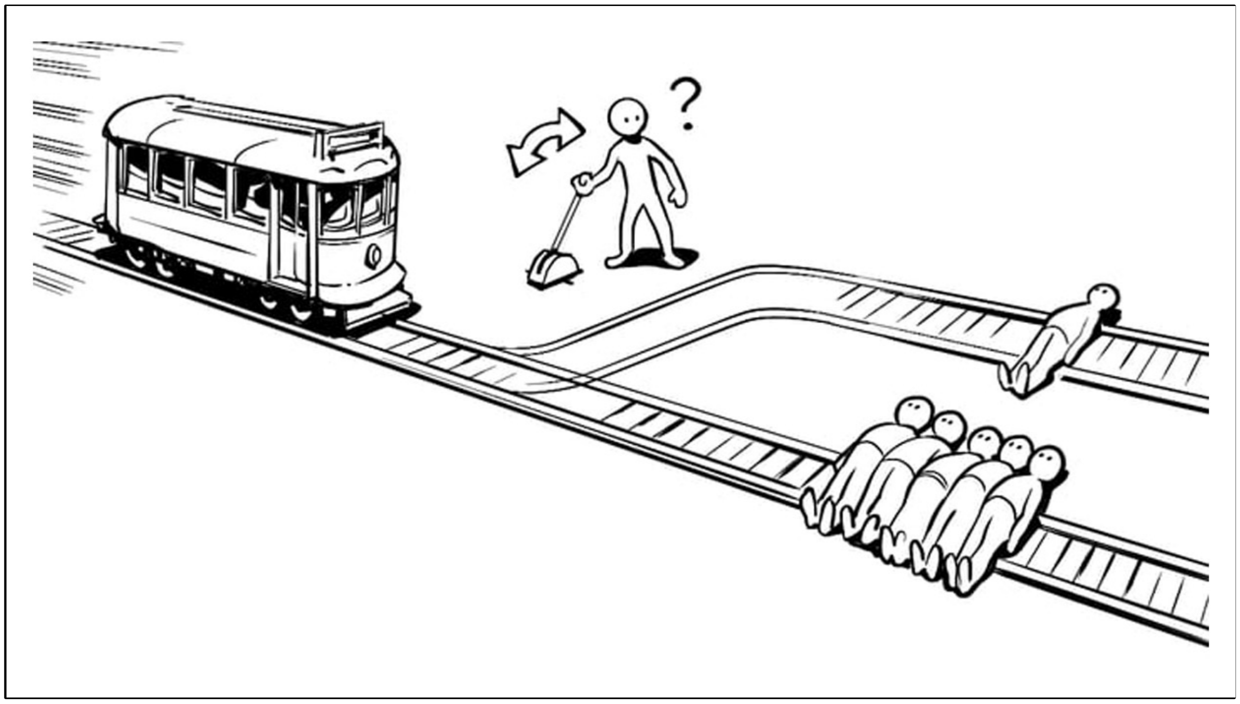
you founding a new religion that kills a guy while his wife drinks animal blood

LIABILITY

Or you getting sued and not being able to defend yourself effectively.

keeping a human in the loop is important for another, slightly darker reason: it gives you someone to blame when things go wrong.

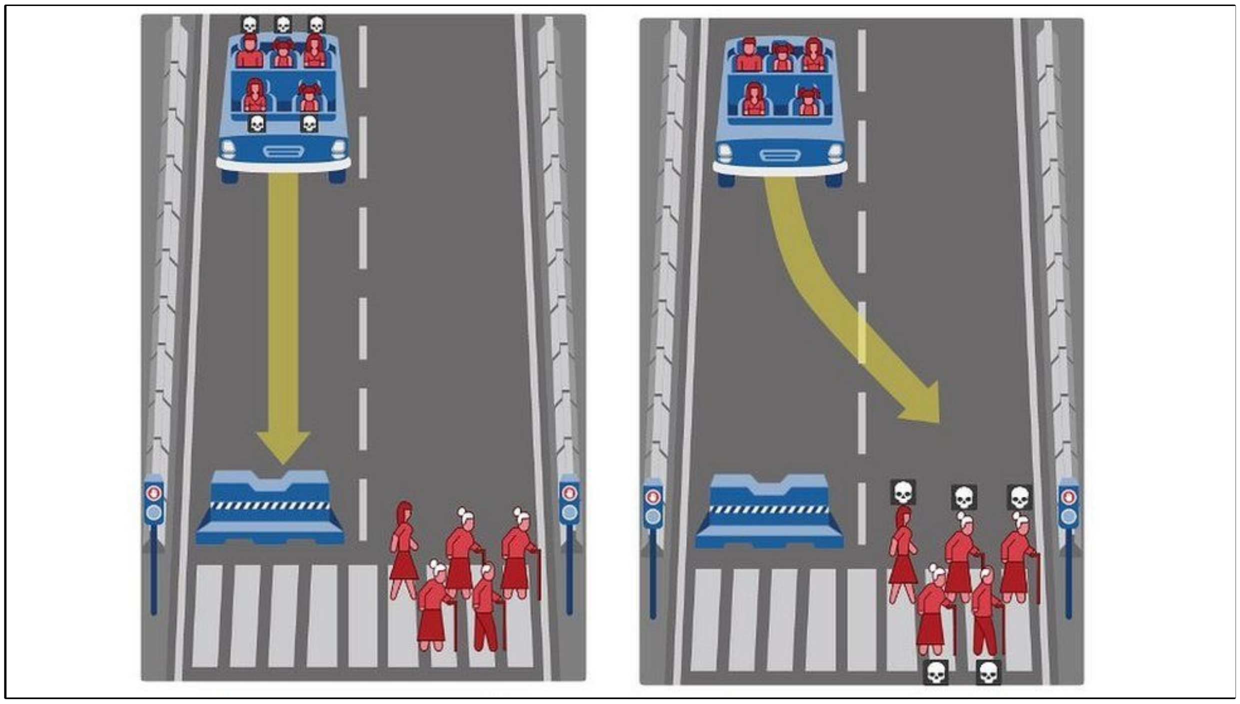
let's talk about the trolley problem.



you probably already know about it — imagine you're a bystander and a trolley is on the tracks ahead of you.

it's running down the tracks and about to hit a big group of people, but you can pull the switch so it hits one person instead.

Do you do it?



let's take this and apply it to self-driving cars.

the car is in a situation where it's about to hit a wall at high speed — high enough that the driver is certain to die when it happens.

but it could swerve out of the way and barrel into a crowd of people crossing the street instead.

this would kill and maim them, but the driver would survive.

what should it do?

what should it be _programmed_ to do?

and when a self-driving car is involved in an inevitable fatal accident, who's liable for it?

is it the car company? the person who trusted the car not to kill them?

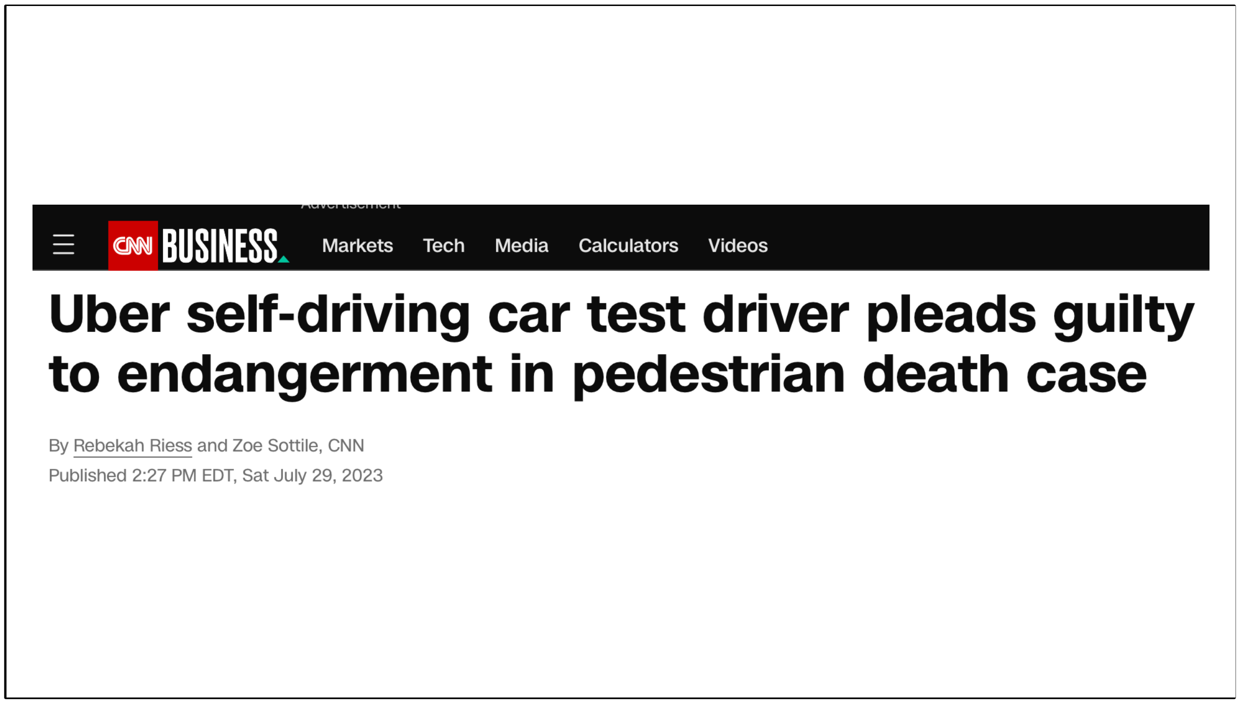
is it the person who designed the algorithm?

now, i am not a lawyer or a philosopher, but i think we can all tell that this is murky ground with no good, easy answers.

and there aren't really any answers out there right now.

there have been a few cases with self-driving cars killing people, but those weren't truly autonomous — they had someone sitting in the driver's seat who was supposed to take over.

there was a human to blame — and guess what, that human got blamed and went to prison for it.



Uber self-driving car test driver pleads guilty to endangerment in pedestrian death case

By Rebekah Riess and Zoe Sottile, CNN
Published 2:27 PM EDT, Sat July 29, 2023

and there aren't really any answers out there right now.

there have been a few cases with self-driving cars killing people, but those weren't truly autonomous — they had someone sitting in the driver's seat who was supposed to take over.

there was a human to blame — and guess what, that human got blamed and went to prison for it.

when it comes to medical, we usually say the buck stops with the physician.

we make all sorts of algorithmic recommendations and provide tons of feedback,

but at the end of the day we absolve ourselves of guilt by requiring a doctor to sign off.

that signature is a transfer of liability from us, the designers and company, to the doctor and their malpractice insurance.

TELLING IS NOT ENOUGH

there is an assumption that if we tell our users what the risks are,

what percentage of false positives and negatives there are,

they make rational decisions and accept that risk to themselves.

"5% false negative rate," we say, and the physician is implicitly comfortable with the idea.

but how much of that information is really communicated, and how much is hidden away in tiny text?

when was the last time you saw a system advertised that said ”with our algorithm, only a million people every year are falsely cleared from having cancer!"

we, as human beings, are really bad at assessing risk and thinking about the consequences of it.

Right now there is basically no precedent for any of this.

But surely the day is coming where a patient dies because an algorithm cleared them incorrectly.

there will be court cases,

with a physician blaming the system manufacturer and

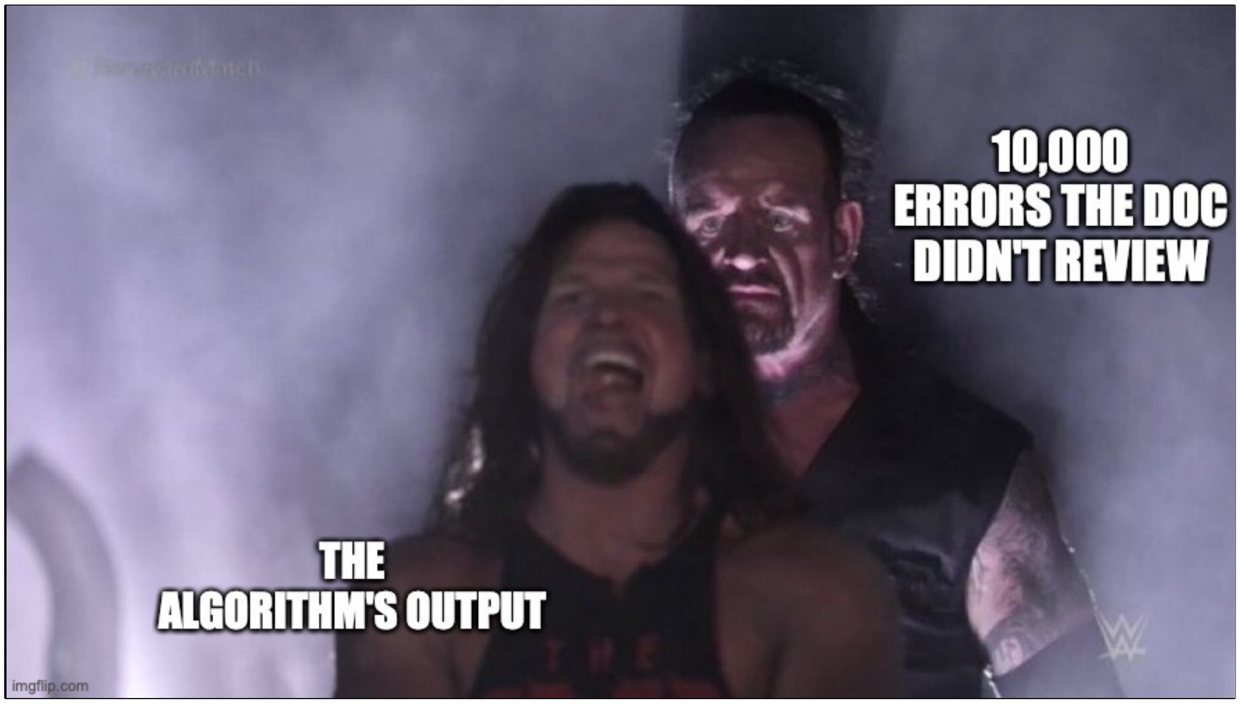
the company blaming the physician for not paying close enough attention.

but where is the line?

if you keep all the outputs the algorithm produced, is that better than deleting them?

what if you have 1000 negative results, and just five of them are actually false negatives?

is it reasonable to ask your customers to have pored over that data themselves to find it?



as algorithms become more common,

we risk overloading our users with data that obscures the real risks to a patient.

algorithms are intending to help us sift signal from noise,

but the ironic thing is that we may just be creating new noise to help us hedge our bets.

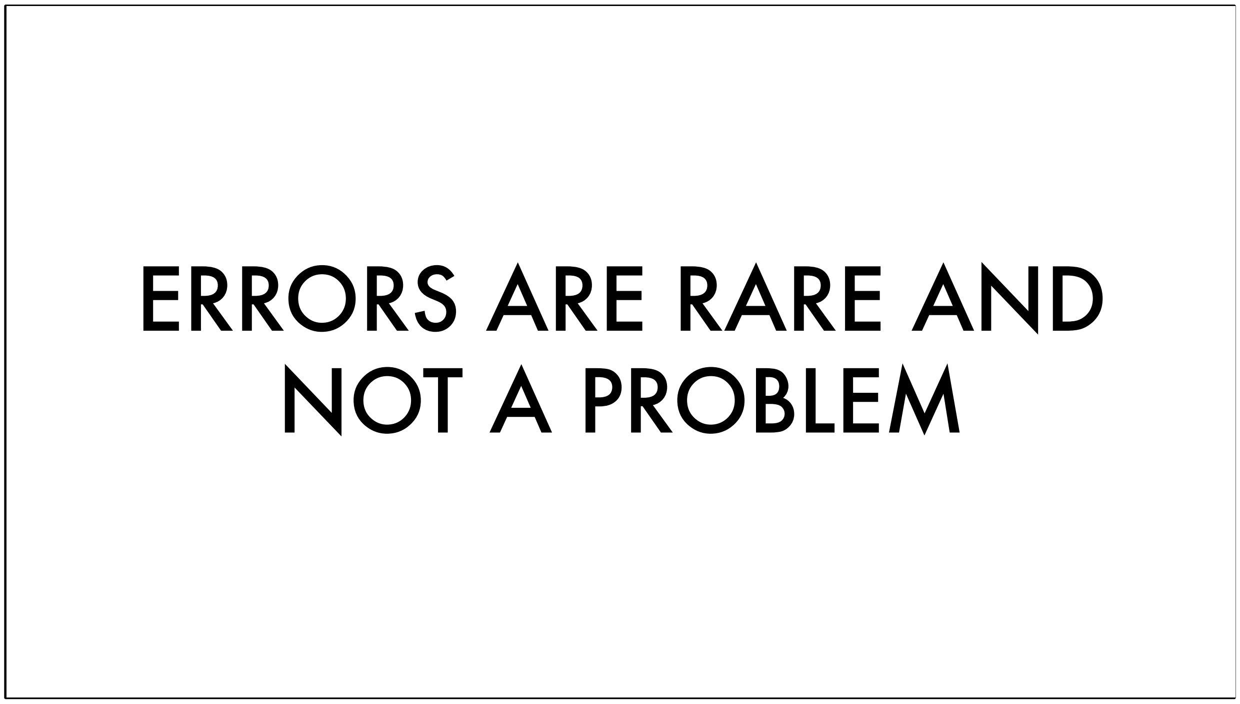
we have a responsibility as designers to make sure that we're not just giving users access to _all_ the data,

but to the _right_ data. and we need to make sure that we help them have insights into not just what an algorithm is doing,

but what it produced. we need to make sure they understand what a 5% risk is, what a 10% risk is.

what does that mean for their patients? what does it mean for _them?_

who wants to go to prison because an algorithm screwed up?

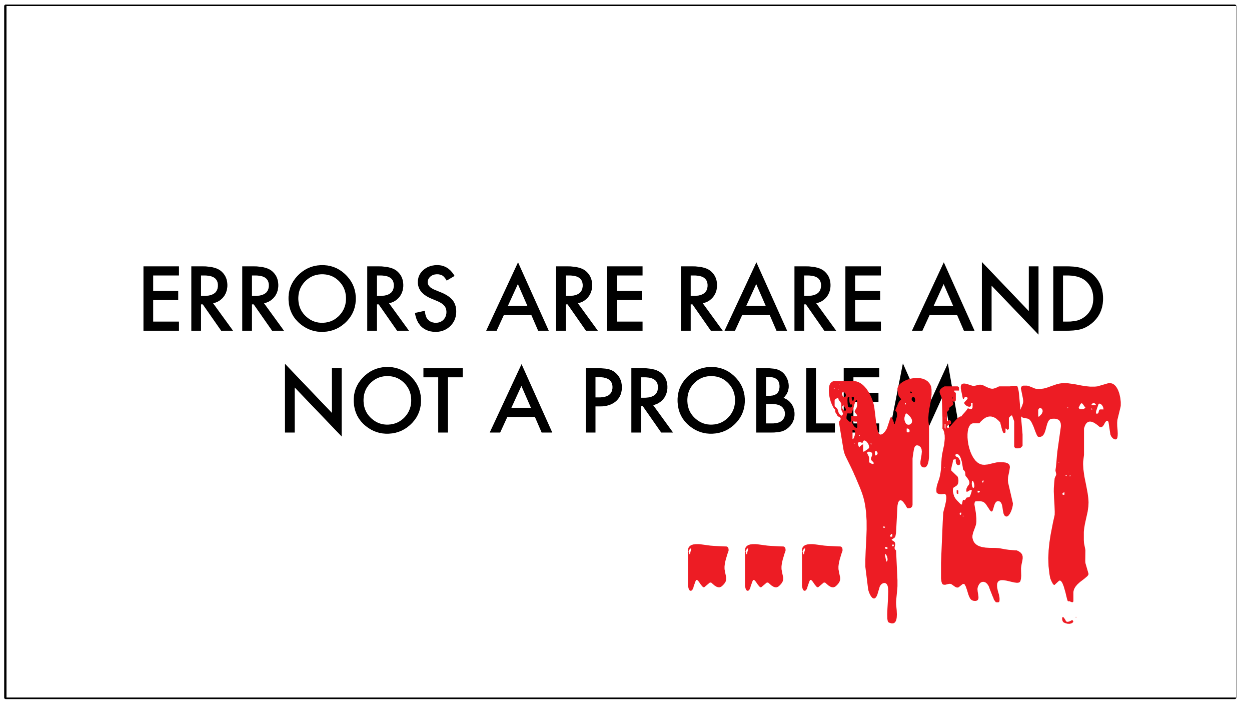


ERRORS ARE RARE AND NOT A PROBLEM

surely people have been harmed, but i suspect that until now, it's mostly been buried in a bunch of other medical and legal data that obscures the root cause.

right now a lot of us are operating in a space where these aren't concerns because the risks are low. errors "just don't happen.” or they aren’t a big deal

but i'd like to amend that.

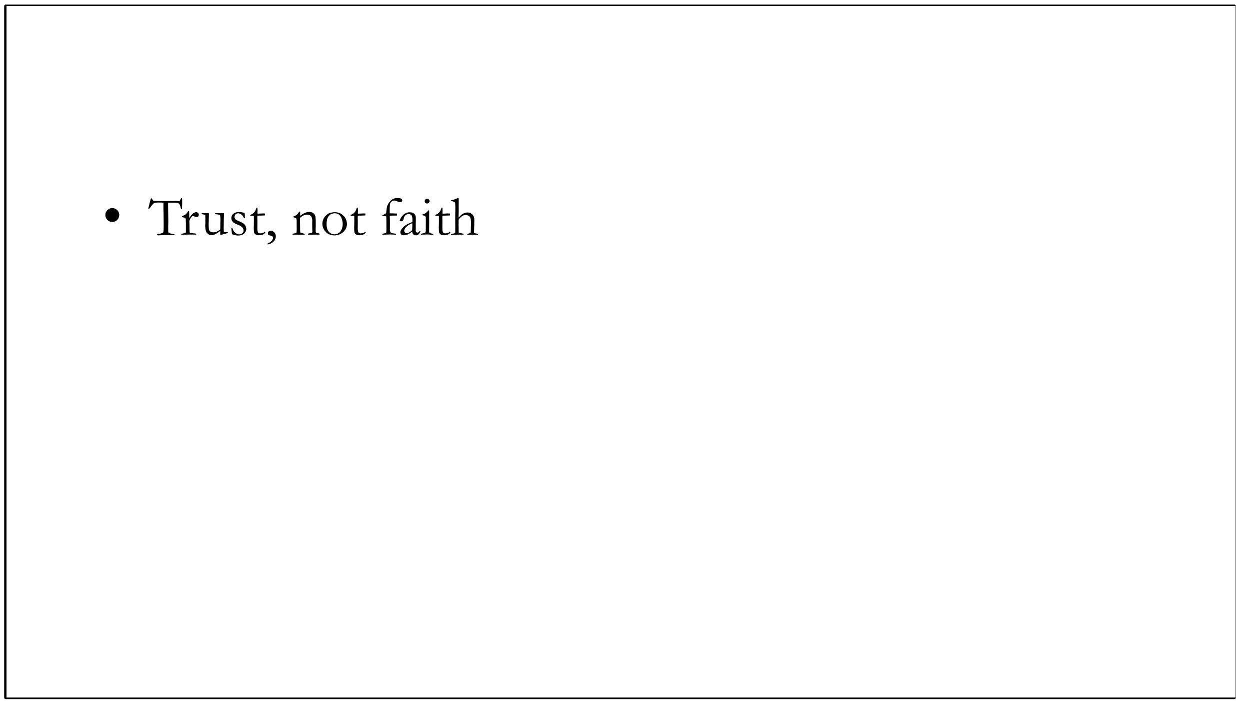


ERRORS ARE RARE AND NOT A PROBLEM

...YET

error's "just don't happen — yet."

mark my words, the day is coming where an algorithm is to blame, and when it does, it will be a seismic shift in how we use them.



- Trust, not faith

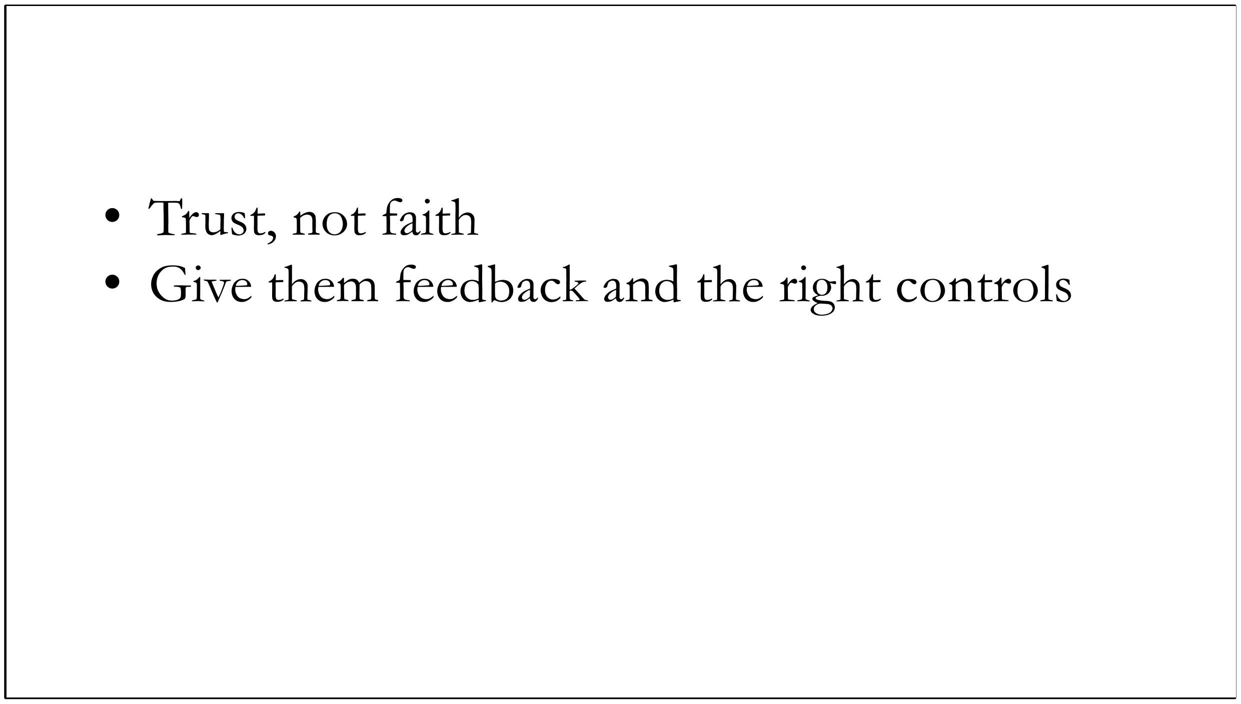
here's what i want to leave you with today, four key things to think about when building interfaces for algorithms.

Trust, not faith. No black boxes!

Show them how it works, and give them access to the levers that suit their need and their ability level

Tell them about errors the system can notice and give them tools to look into surprising results

And don’t be Aleister Crowley!



- Trust, not faith
- Give them feedback and the right controls

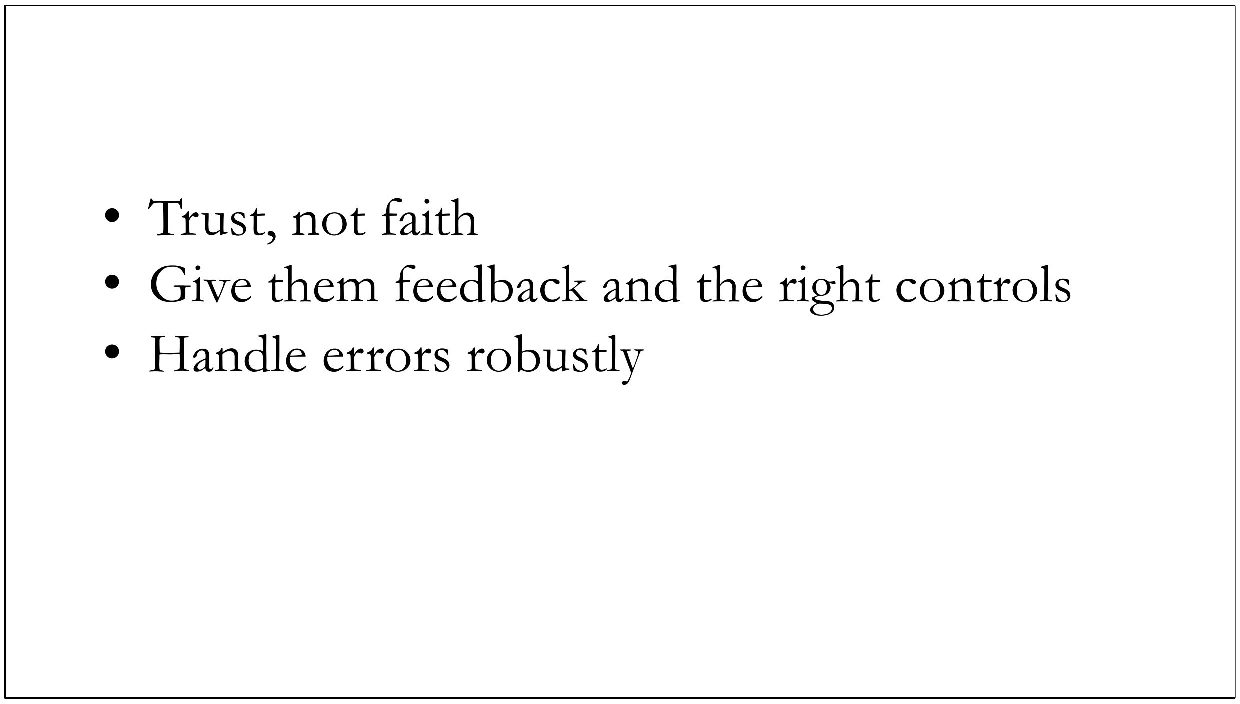
here's what i want to leave you with today, four key things to think about when building interfaces for algorithms.

Trust, not faith. No black boxes!

Show them how it works, and give them access to the levers that suit their need and their ability level

Tell them about errors the system can notice and give them tools to look into surprising results

And don’t be Aleister Crowley!



- Trust, not faith
- Give them feedback and the right controls
- Handle errors robustly

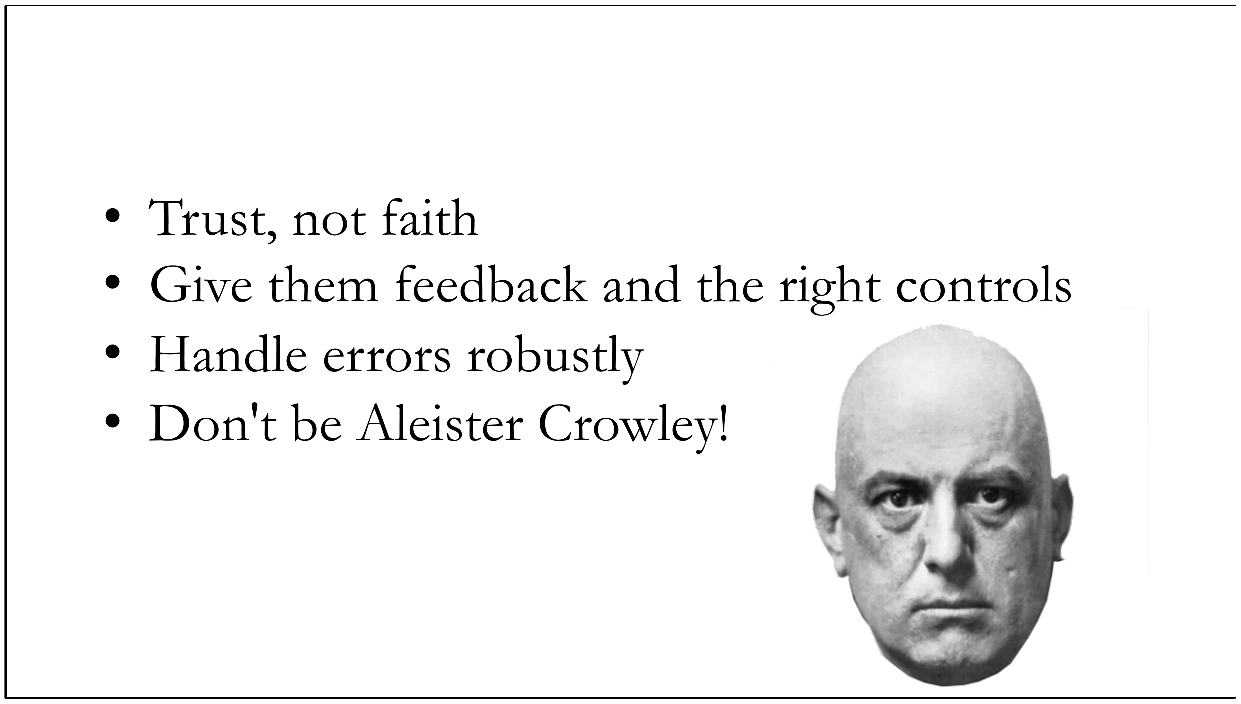
here's what i want to leave you with today, four key things to think about when building interfaces for algorithms.

Trust, not faith. No black boxes!

Show them how it works, and give them access to the levers that suit their need and their ability level

Tell them about errors the system can notice and give them tools to look into surprising results

And don’t be Aleister Crowley!



- Trust, not faith
- Give them feedback and the right controls
- Handle errors robustly
- Don't be Aleister Crowley!

here's what i want to leave you with today, four key things to think about when building interfaces for algorithms.

Trust, not faith. No black boxes!

Show them how it works, and give them access to the levers that suit their need and their ability level

Tell them about errors the system can notice and give them tools to look into surprising results

And don’t be Aleister Crowley!