HACKERNOON

Machine learning — Is the emperor wearing clothes?

sie
yrkov
zyrkov

September 14th 2018

TWEET THIS

A behind-the-scenes look at how machine learning works

Machine learning uses patterns in data to label things. Sounds magical? The core concepts are actually embarrassingly simple. I say "embarrassingly" because if someone made you think it's mystical, they should be embarrassed. Here, let me fix that for you.

> The core concepts are embarrassingly simple.

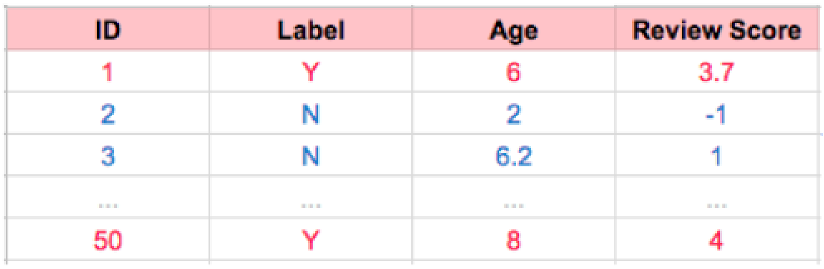
Our thing-labeling example will involve classifying wine as yummy or not-so-yummy and we'll keep all the ideas simple enough to enjoy alongside a glass of wine… or three. If wine is not your cup of tea, here's an alcohol-free version of the same text.

How does it work?

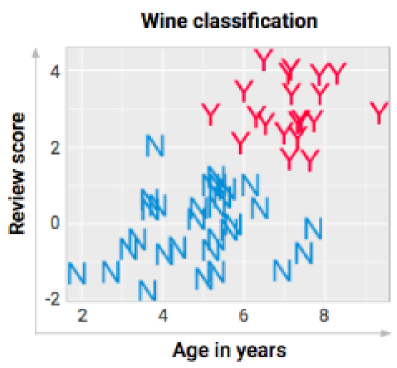
ML's not magic—it's impossible to learn without data, so I'll have to taste some wine. If you must know, this one got a N label, N for nope-let's-not-try-this-again . And I got it all over myself. The things we do for science.

Data

To learn, you need something to learn *from*. Let's imagine I tasted 50 wines (for science!) and visualized their info for your viewing pleasure below. Each wine has an age in years and a review score, plus the correct answers we're trying to learn: Y for yummy and N for not-so-yummy.



ID	Label	Age	Review Score
1	Y	6	3.7
2	N	2	-1
3	N	6.2	1
...	...	...	...
50	Y	8	4



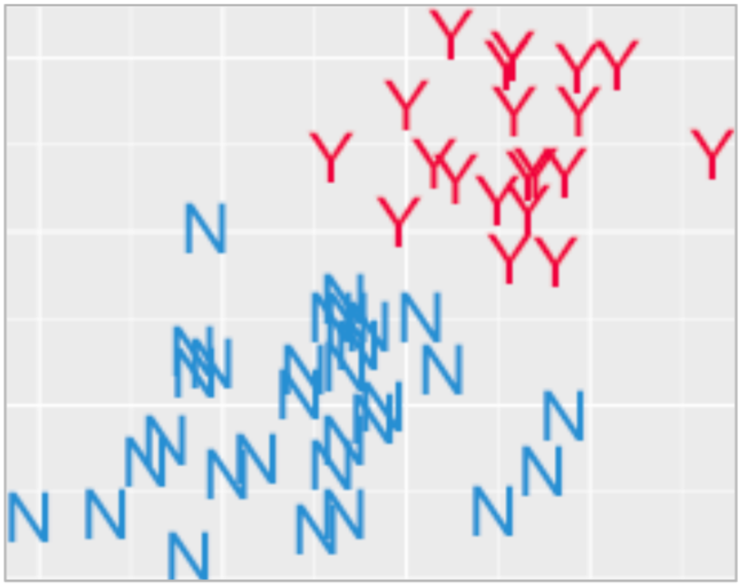
After I've tasted the wines and recorded their data in a spreadsheet (left), good manners dictate that I show the info to you in a more eye-friendly format (right). If you're craving dataset jargon like features and instances, here's my guide to that stuff. Since we're among friends, "inputs" will do just fine.

Algorithm

By picking a machine learning algorithm to use, we're picking the type of recipe we're going to get. Why don't you be my algorithm? Your entire job is to separate the red things from the blue. Can you do it?

> The purpose of a machine learning algorithm is to pick the most sensible place to put a fence in your data.

If you thought about drawing a line, congratulations! You just invented a machine learning algorithm whose name is... *perceptron*. Yeah, such a sci-fi name for such a simple thing! Please don't be intimidated by jargon in machine learning, it usually doesn't deserve the shock and awe the name inspires.



How would you cordon off the red things from the blue?

But ***where*** does your line go? I hope you'd agree that a flat line is not a very smart solution. Our goal is to separate Y from N, not decorate the horizon.

The purpose of a machine learning algorithm is to pick the most sensible place to put the fence, and it decides that based on where your datapoint have landed. How does it do that? By *optimizing* an objective function.

Optimization

I plan to give optimization its own blog post, but for now think of it like this: the objective function is like the rule for scoring a board game, optimizing it is figuring out how to play so that you earn the best score possible.

An objective function (loss function) is like the point system for a board game. This photo suggests that back in college I had yet to learn optimization… why am I playing the land-war-in-Asia strategy?

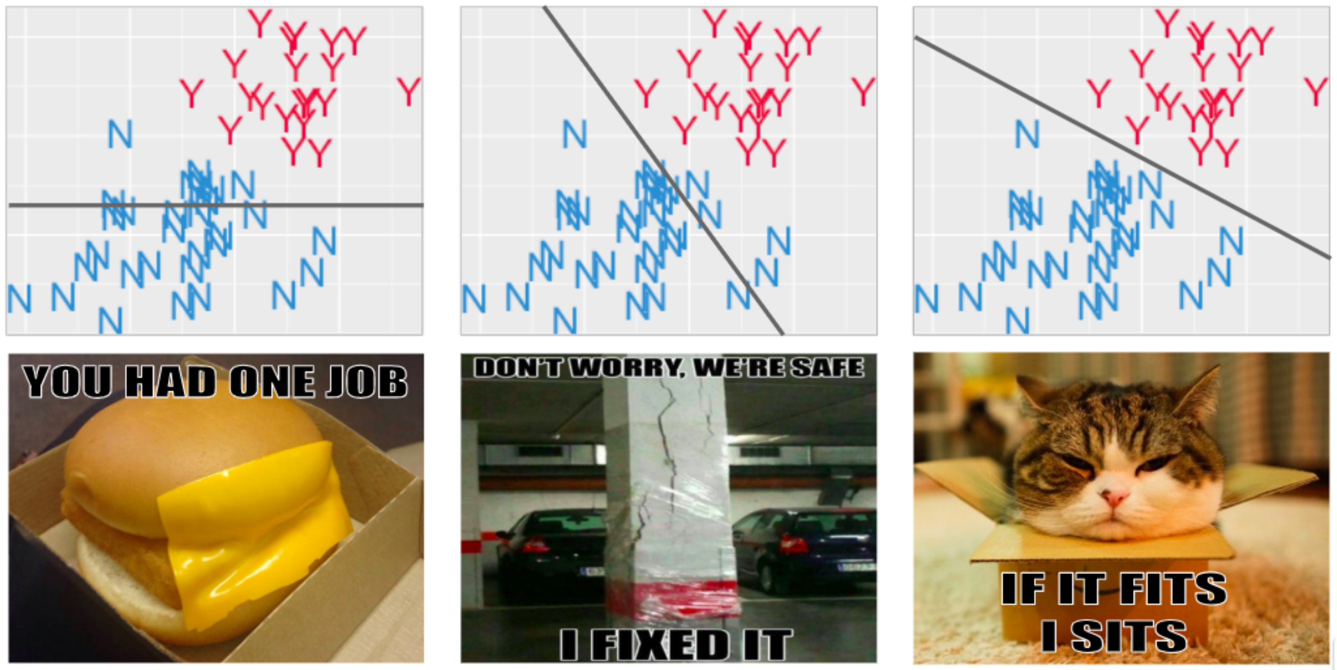
Traditionally in ML, we like sticks more than carrots—the points are penalties for mistakes (getting a label on the wrong side of the fence) and the game is to get as few of these bad points as possible. That's why the objective function in ML tends to be called a "loss function" and the goal is to minimize loss.

> The loss function is like the rule for scoring a board game, optimizing it is figuring out how to play so you get the best score possible.

Want to play? Go back to the plot above, put your finger horizontally against the screen and rotate it until you get a score of zero bad points (no points escaping the segregating wrath of your mighty finger). Congratulations, Comrade Perceptron! Feeling futuristic yet?

> Jargon in machine learning usually doesn't deserve the shock and awe the name inspires.

The solution you came to is hopefully something like this:



In the leftmost image, c'mon; we're not even trying. Hopefully you'll agree that the middle one is better, but it still doesn't fit as well as it can. I'm a fan of rightmost one.

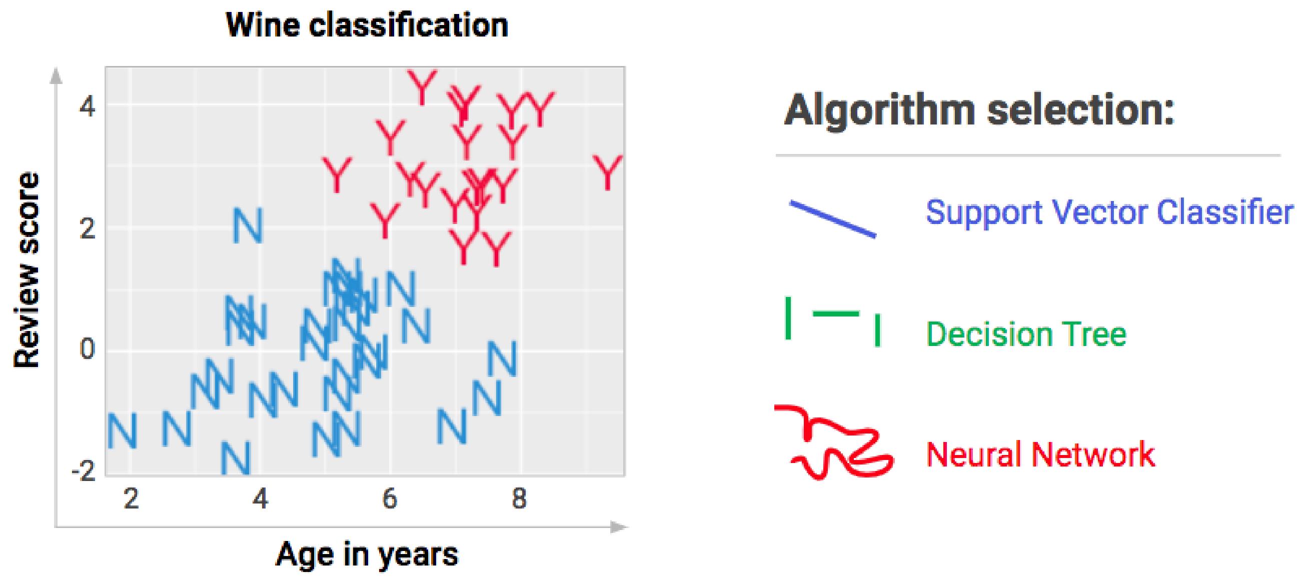
The spice of life

If you like variety, you'll love algorithms. There are plenty of them. One way they differ from one another is in how they try on different positions for the separating boundary.

> If you like variety, you'll love algorithms. There are so many!

Optimization nerds will tell you that rotating the fence in tiny increments is certifiably lame (sorry for tricking you into it!) and there are much better ways to get to the optimal position faster. Some researchers devote their whole lives to coming up with ways to get the best fence position in fewest hops, no matter how perverse the terrain (determined by your inputs) becomes.

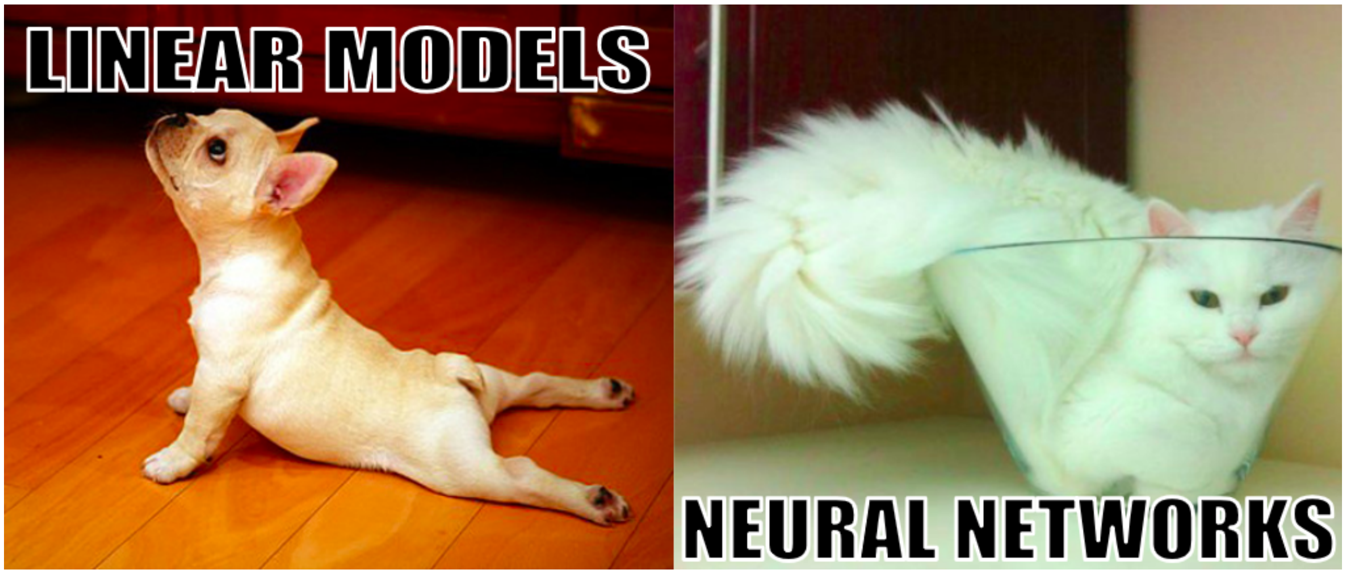
Another source of variety is the shape of the boundary. Turns out the fence needn't be straight. Different algorithms use different kinds of fences.



When we pick these gobbledygook names, we are simply picking the shape of the boundary that we are drawing between the labels. Do we want to separate them with one diagonal line or many horizontal/vertical lines or flexible squigglers... or something else? There are lots and lots of algorithms to pick from.

Algorithms for hipsters

These days, no data science hipster is into the humble straight line. Flexible, squiggly shapes are all the rage among today's fashionable crowd (you may know these as *neural networks*, though there isn't much that's neural about them—they were named rather aspirationally more than half a century ago and no one seems to like my suggestion that we rename them to "yoga networks" or "many-layers-of-mathematical-operations").



Instead of the marketing-hype-scented comparison of other algorithms with lines and neural networks with brains, it's better to think of them in terms of contortionist ability. Those other methods are simply less talented at data yoga. Nothing is free, however, and neural networks exact a hefty price (more on this soon!), so don't trust anyone who tells you they're always the best solution.

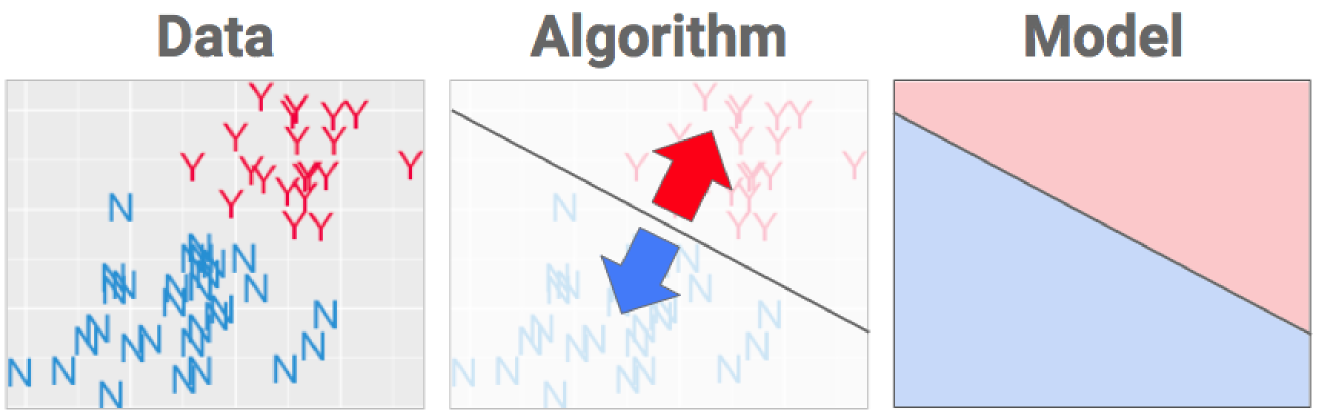
> *Neural networks may as well be called "yoga networks"—their special power is giving you a very flexible boundary.*

Those gobbledygook algorithm names tell you what ***shape*** of fence they're going try to put in your data. If you're an applied machine learning enthusiast, it's okay if you don't memorize them—in practice you'll just shove your data through as many algorithms as you can and iterate on what seems promising. The proof of the pudding's always in the eating… so let's eat.

Even if you study the textbook, you won't get the solution right on your first try. Don't sweat it. This isn't a game with one right answer and no one gets their solution on the first try. Give yourself permission to tinker, dabble, and play. The proof of machine learning pudding's in the eating—does it work on new data? Leave the "***how*** does it work" for researchers who design new algorithms. (And you'll probably end up getting familiar with those names the same way you learned the characters of whichever bad soap opera haunts your town's TV screens. Against your will. Give it time.)

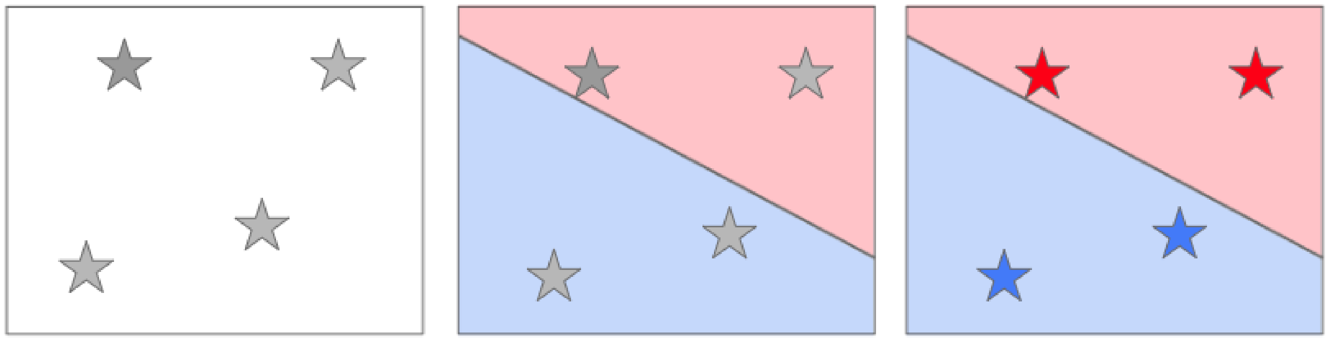
Model

Once the fence in place, the algorithm is finished and what you get out of it is what you wanted along: a model, which is just a fancy word for recipe. It's now some instructions for the computer to use to convert data into a decision next time I show it a new bottle of wine. If the data lands in the blue bit, call it blue. In the red bit? Call it red.



Label

Once you put your freshly-minted model into production, you use it by feeding the computer an age and review score. Your system looks up which region that corresponds to and it outputs a label.



When I get four new bottles of wine, I simply match their input data against the recipe's red and blue regions and label them accordingly. See? It's easy!

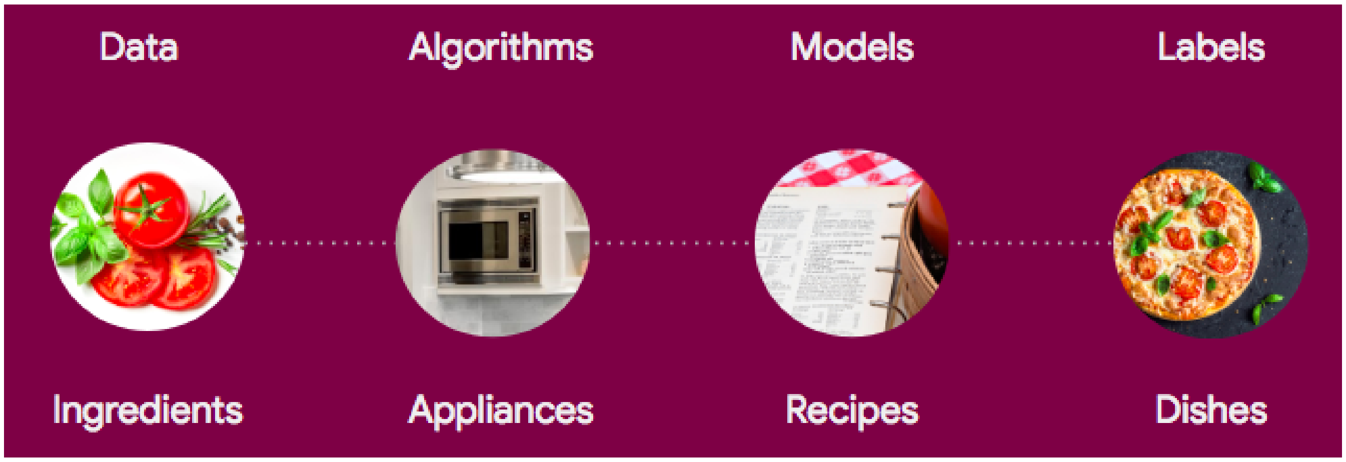
How do we know if it works? The same way we know if a bunch of monkeys on typewriters wrote some Shakespeare. By checking the output!

> The proof of the pudding is in the eating.

Test your system by running a bunch of new data through it and make sure it performs well on it. In fact, do that anyway, regardless of whether an algorithm or a programmer came up with that recipe for you.

Summary

Here's a handy visual summary for you from another of my articles:



Machine learning for poets

If that was confusing, maybe you'll like this analogy more: A poet picks an approach (algorithm) to putting words on paper. The approach determines the resulting poetic form (boundary shape)—will it be a haiku or a sonnet? Once they're finished optimally fleshing out their sonnet, it is now a poem (model). How is a poem a recipe? Beats me—I'm open to suggestions. The rest of the analogy works, though.

ML models vs traditional code

I'd like to point out that the recipe isn't all that different from code that a programmer might have written by eyeballing the problem and manually making up some rules. Quit anthropomorphizing machine learning already. A model is conceptually the same kind of thing as regular code. You know, the kind of recipe that's handcrafted by some human armed with an opinion and a caffeine source.

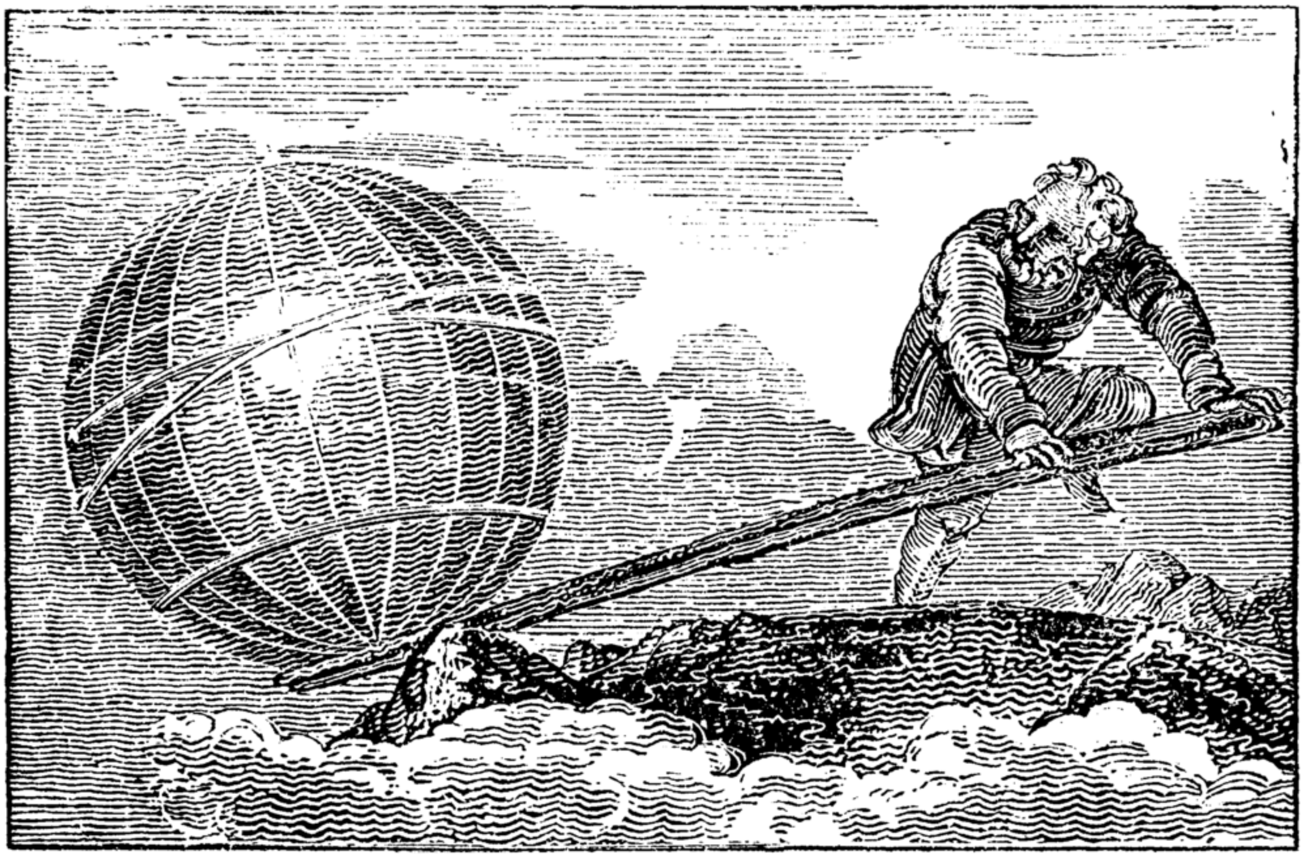
> Quit anthropomorphizing machine learning already.

And don't go around saying that retraining—jargon for rerunning the algorithm to adjust the boundary as new examples are gathered—makes it creature-like or inherently different from your programmer's standard work product. Humans can sit there tweaking the code in response to new info too. If you're worried that your machine learning system now does the update much faster, invest in good testing or avail yourself of a long *time.sleep()*.

Is this all there is to it?

Yeah, pretty much. The hard part of ML engineering is installing packages and then wrangling your hairy beast of a dataset so a finicky algorithm deigns to run on it. This is followed by an eternity of fiddling with the code settings (don't let the noble name *"hyperparameter tuning"* fool you) until voilà! A model! One that turns out not to work when you evaluate its performance on new data… and you're back to the drawing board, over and over, until finally the heavens open up and your solution stops embarrassing itself. That's why it's so important to hire failure-tolerant personalities for this endeavor.

> Don't hate it for being simple. Levers are simple too, but they can move the world.



If you were expecting magic, well, the sooner you're disappointed, the better. The death of superstitious excitement for sci-fi makes way for rebirth: excitement for getting cool things done. Machine learning may be prosaic, but what you can do with it is incredible! It lets helps you write the code you couldn't come up with yourself, allowing you to automate the ineffable. Don't hate it for being simple. Levers are simple too, but they can move the world.

If you're keen to get started with an AI/ML project, I've got a step-by-step guide for you here. Enjoy!

Machine Learning # Artificial Intelligence # Decision Intelligence # Data Science # Technology

Continue the discussion

More by Cassie Kozyrkov

9 Things You Should Know About TensorFlow

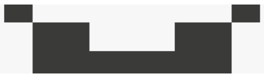
Cassie Kozyrkov

# Machine Learning

Pay attention to that man behind the curtain

Cassie Kozyrkov

# Machine Learning



Is your AI project a nonstarter?

Cassie Kozyrkov

# Machine Learning

Are you using the term 'AI' incorrectly?

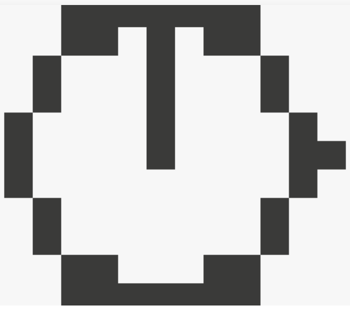
Cassie Kozyrkov
May 26

# Machine Learning

Best Data Science Overviews

Cassie Kozyrkov
Jun 05

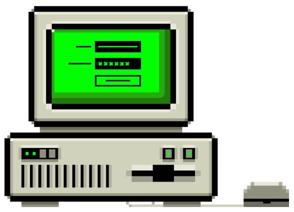
# Hackernoon Top Story



Data-Driven? Think again

Cassie Kozyrkov

# Data Science



Hackernoon Newsletter curates great stories by real tech professionals

Get solid gold sent to your inbox. Every week!

Email Address *

First Name

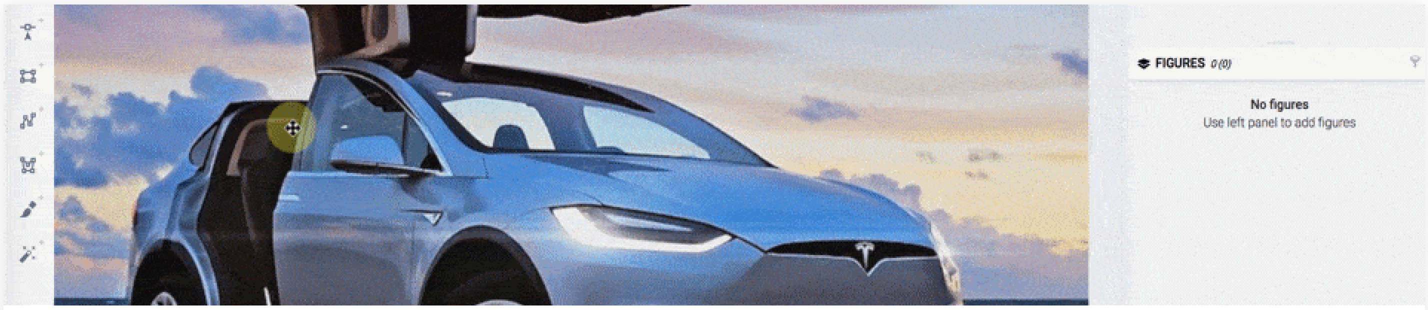
Last Name

TOPICS OF INTEREST

✓ Software Development	✓ Blockchain Crypto
✓ General Tech	✓ Best of Hacker Noon

Get great stories by email

More Related Stories



✏ Advanced annotation tools in Deep Learning: training data for computer vision with Supervisely

Supervise.ly
Dec 19

# Saas

1—Cool things this week

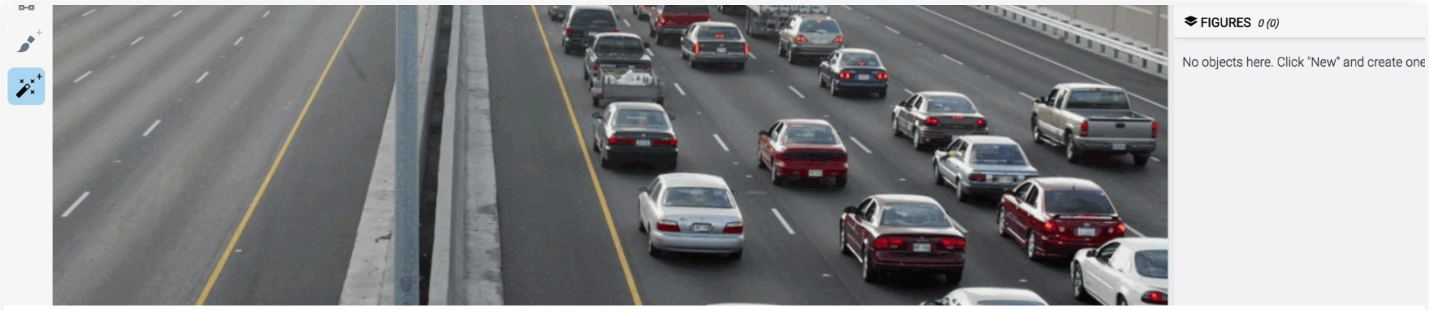
Shreya Amin
Oct 05

# Artificial Intelligence

Explainable AI won't deliver. Here's why.

Cassie Kozyrkov
Nov 16

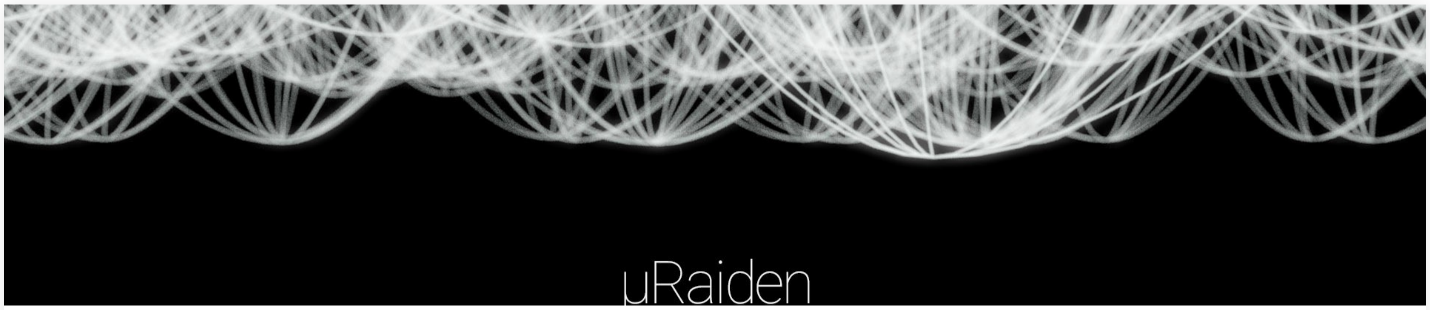
# Artificial Intelligence



⚔ Big challenge in Deep Learning: training data

Deep Systems
Nov 21

# Data Science



µRaiden: Micropayments for Ethereum

Raiden Network
Sep 19

# Ethereum