Machine Learning in Production: Why You Should Care About Data and Concept Drift

No model lasts forever. Even if the data quality is fine, the model itself can start degrading. What does this mean in practice?

Elena Samuylova · Follow
Published in Towards Data Science
10 min read · Dec 7, 2020

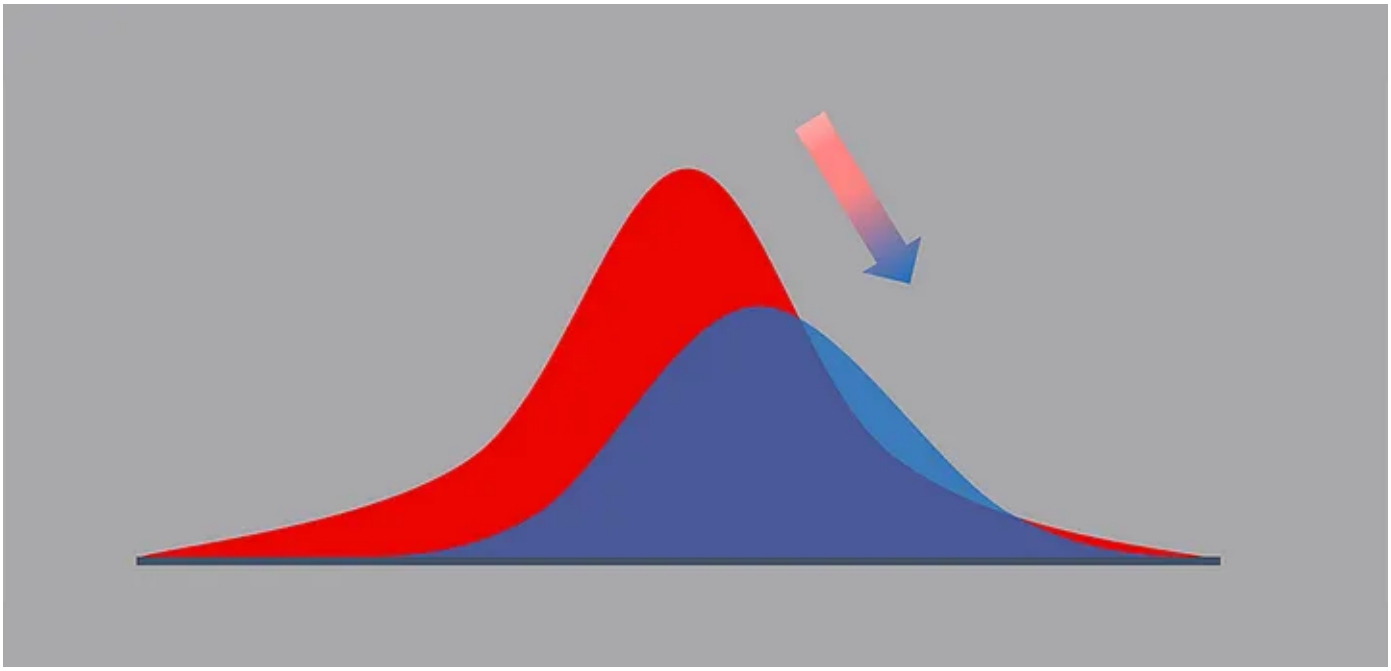


Image by author.

What can go wrong?

Machine learning models often deal with corrupted, late, or incomplete data. Data quality issues account for a major share of failures in production.

But let's say it is covered. The data engineering team does a great job, data owners and producers do no harm, and no system breaks. Does this mean our model is safe?

Sadly, this is never a given. While the data can be right, the model itself can start degrading.

A few terms are used in this context. Let's dive in.

Model decay

> *Past performance is no guarantee of future results. — Small print for most financial products out there.*

People call it **model drift**, **decay**, or **staleness**.

Things change, and model performance degrades over time. The ultimate measure is the model quality metric. It can be accuracy, mean error rate, or some downstream business KPI, such as click-through rate.

No model lives forever, but the speed of decay varies.

Some models can last years without an update. For example, certain computer vision or language models. Or any decision system in an isolated, stable environment.

Others might need daily retraining on the fresh data.

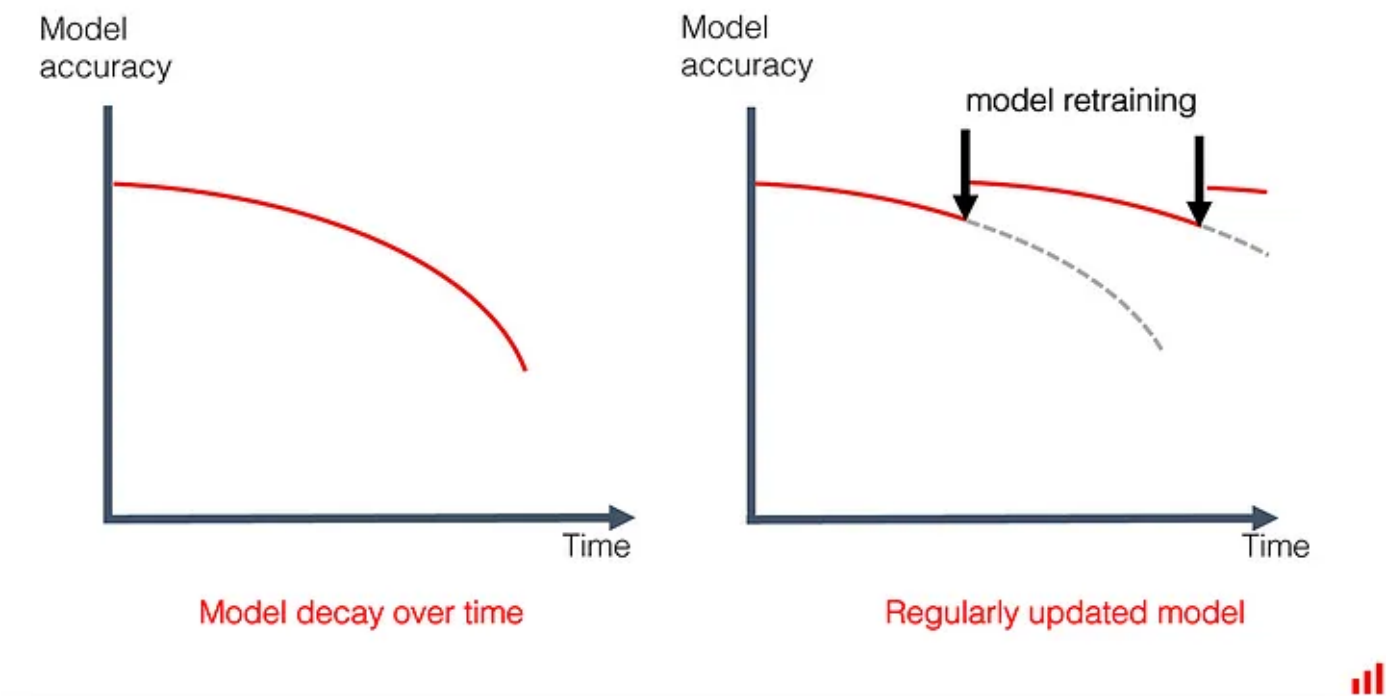


Image by author.

Model decay is a term that literally says that the model got worse. But of course, it happens for a reason.

When data quality is fine, there are two usual suspects: data drift or concept drift. Or both at the same time.

Bear with us. We'll explain it now.

Data drift

Data drift, **feature drift**, **population**, or **covariate shift**. Quite a few names to describe essentially the same thing.

Which is: the input data has changed. The distribution of the variables is meaningfully different. As a result, the trained model is not relevant for this new data.

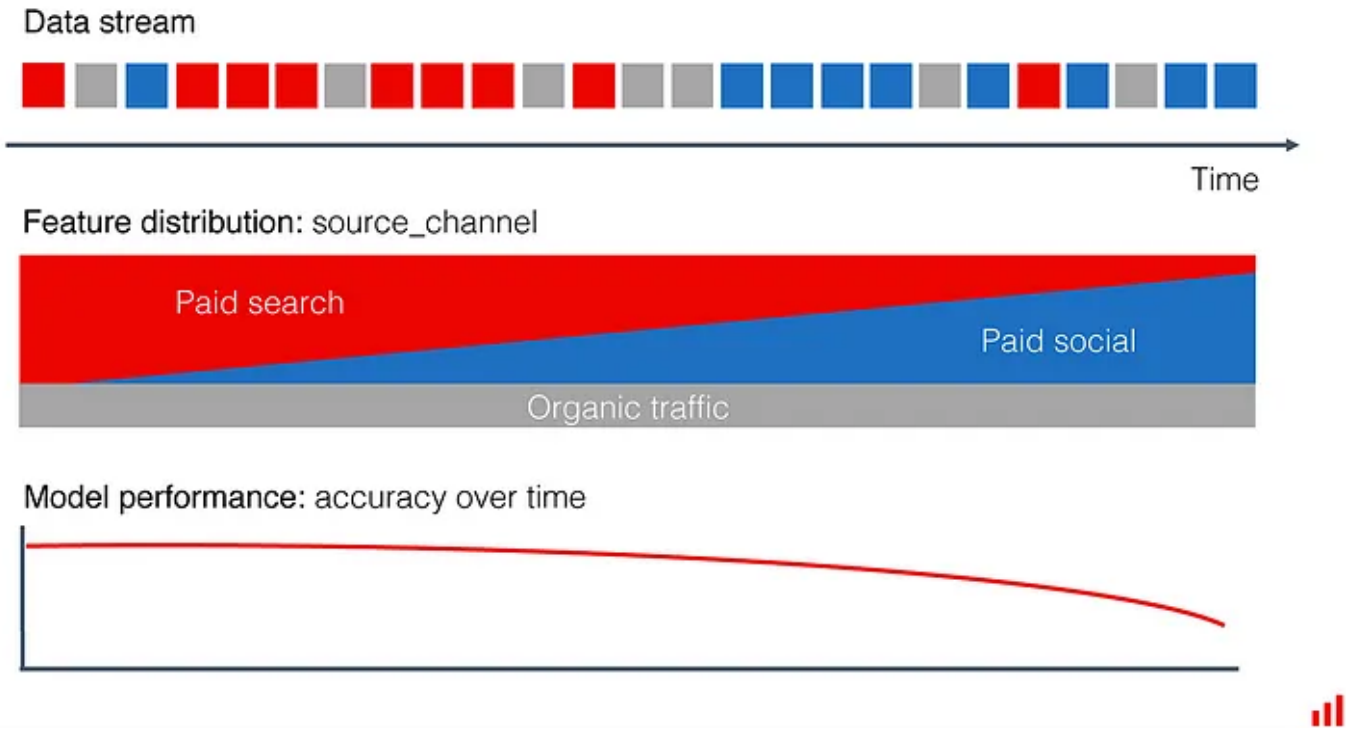
It would still perform well on the data that is similar to the "old" one! The model is fine, as much as the model "in a vacuum" can be. But in practical terms, it became dramatically less useful since we are dealing with a new feature space.

Here is an example of propensity modeling.

An online marketplace attracts users through advertising. Soon after newcomers sign up, we want to predict how likely they will make a purchase in order to send them personalized offers.

Previously most user acquisition happened through paid search. After a new ad campaign, a lot of users come from Facebook. In training, our machine learning model did not do well in this segment. It had limited examples to learn from.

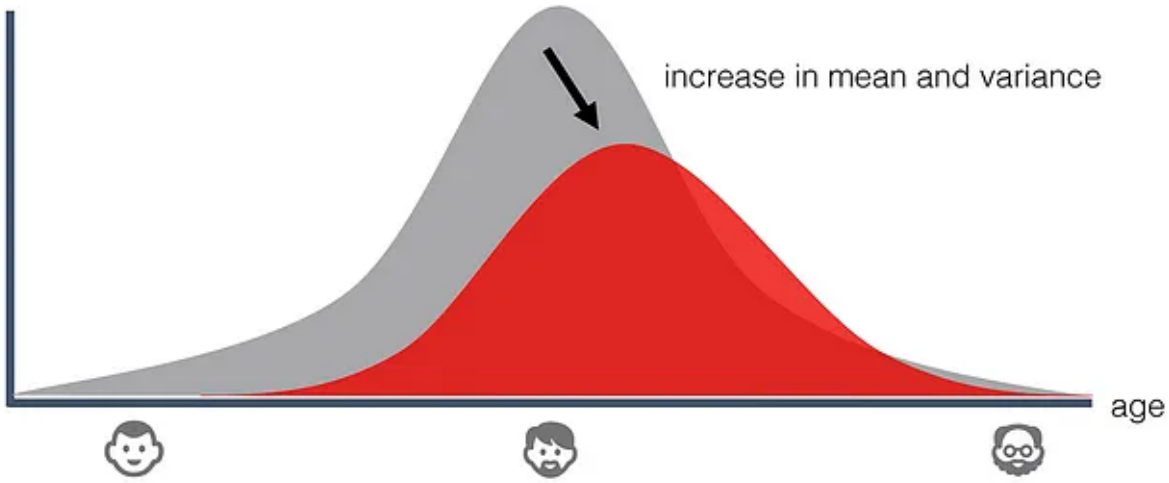
Still, the overall model quality was sufficient since this sub-population was small. Now that it grew, the performance predictably dropped.



As more users come from social channels, the model performance declines. (Image by author).

When debugging, we would see the change in class frequencies in the "source_channel" and distributions of other related variables, such as "device," "region," and so on.

Or, with the monitoring in place, we'd notice this drift early on!



Change in the distribution of the “age” feature. (Image by author).

In these examples, the real-world patterns can remain the same. But the model performance drops as its quality varies for different data regions.

The decay can range from minor to dramatic — when the drift affects the most important features.

To address it, we need to train the model on the new data or rebuild it for the new segment.

Training-serving skew

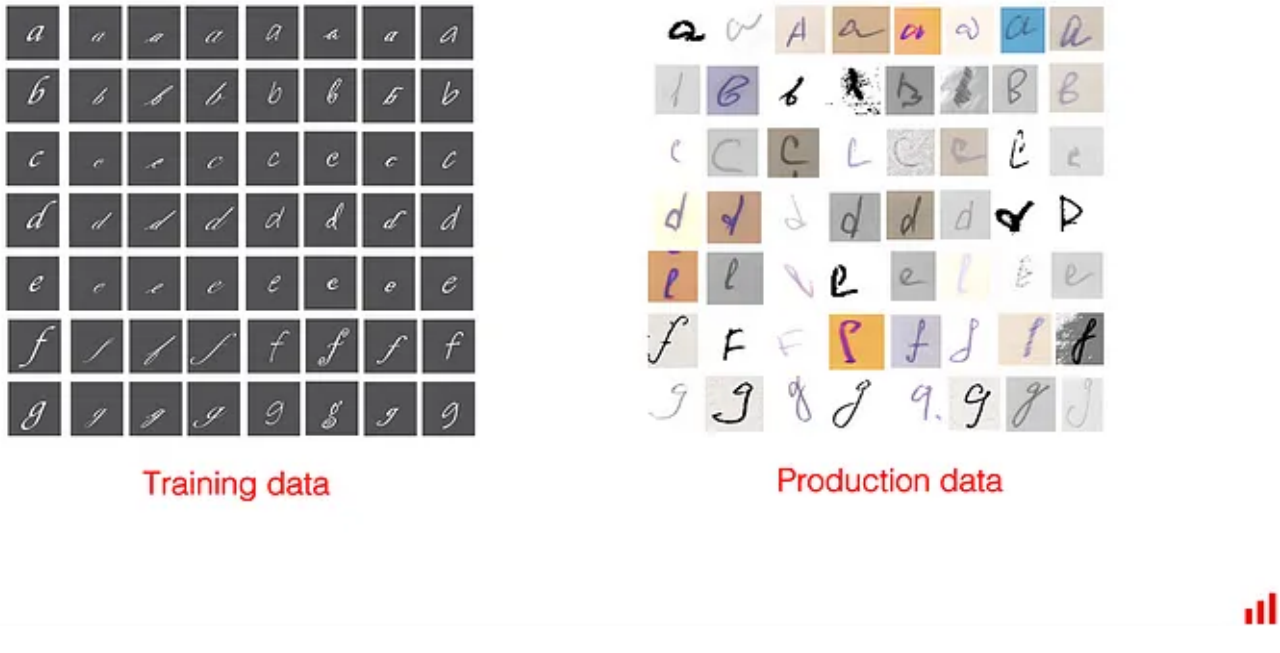
This one deserves an honorable mention.

It is often mixed with data drift or used interchangeably. The way we analyze them might indeed be similar, but the root cause of skew is not the same.

In this case, there is no “drift,” which assumes a change during the production use of the model. Training-serving skew is more of a mismatch. It reveals at the first attempt to apply the model to the real data.

It often happens when you train a model on an artificially constructed or cleaned dataset. This data does not necessarily represent the real world, or does this incompletely.

For example, an invoice classification model is trained on a limited set of crowdsourced images. It does well on the test set. But once put in production, it is surprised by the diversity of how people fill in the invoice data. Or the low quality of scanned images compared to model training.



Training-serving skew. (Image by author).

Recently, the Google Health team faced a similar issue. They developed a computer vision model to detect signs of retinopathy from eye scan images. In practice, those were often taken in poor lighting conditions. The model struggled to perform as well in real life as it did in the lab.

In most cases of training-serving skew, the model development has to continue. If you are lucky, the non-successful trial run might instead generate enough data to train a new model or adapt the existing one. Otherwise, you might need first to collect and label a new dataset.

Concept drift

Concept drift occurs when the patterns the model learned no longer hold.

In contrast to the data drift, the distributions (such as user demographics, frequency of words, etc.) might even remain the same. Instead, the relationships between the model inputs and outputs change.

In essence, the very meaning of what we are trying to predict evolves. Depending on the scale, this will make the model less accurate or even obsolete.

Concept drift comes in different flavors.

Gradual concept drift

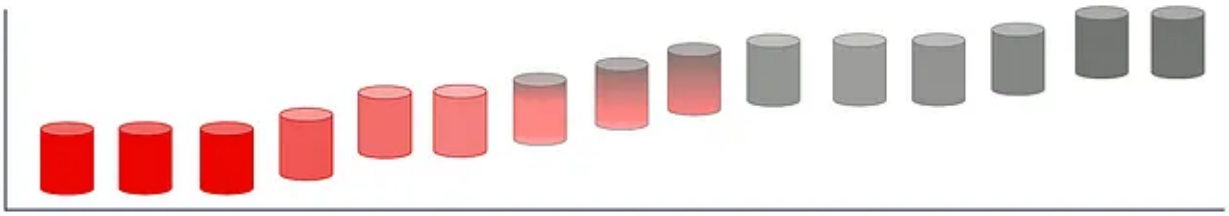


Image by author.

Gradual or **incremental** drift is the one we expect.

The world changes and the model grows old. Its quality decline follows the gradual changes in external factors.

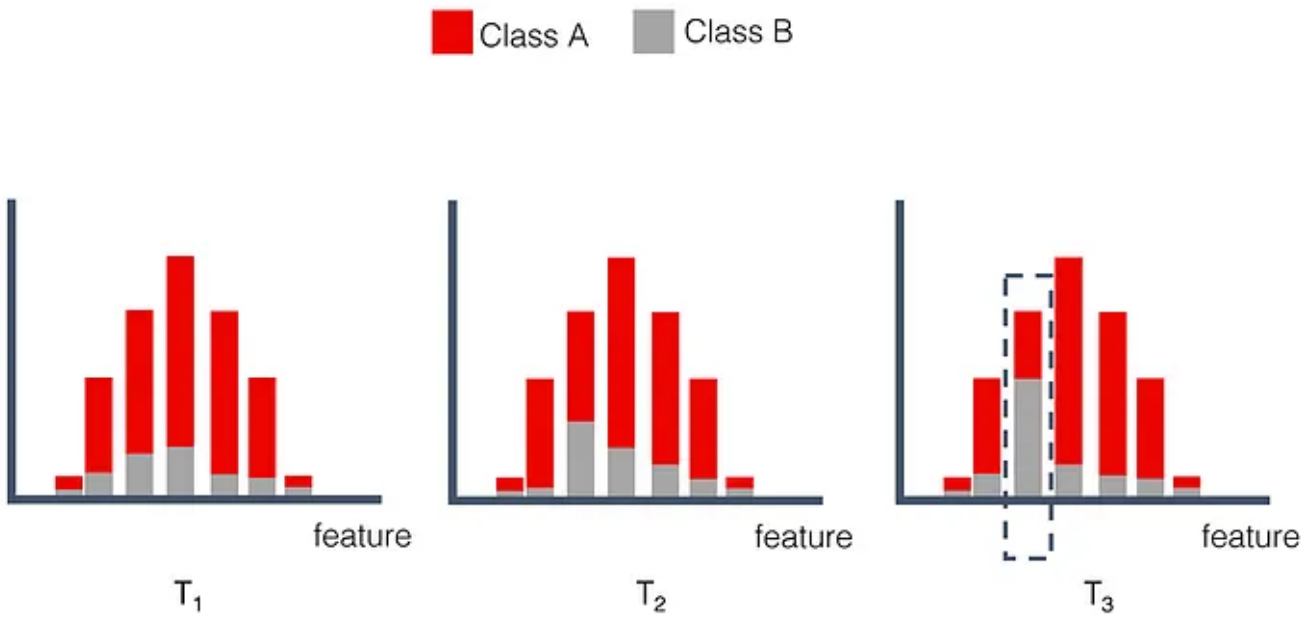
Some examples:

- **Competitors launch new products.** Consumers have more choices, and their behavior changes. As should sales forecasting models.
- **Macroeconomic conditions evolve.** As some borrowers default on their loans, the credit risk is redefined. Scoring models need to learn it.
- **Mechanical wear of equipment.** Under the same process parameters, the patterns are now slightly different. It affects quality prediction models in manufacturing.

No individual change is dramatic. Each might affect only a tiny segment. But eventually, they add up.

Sometimes it is possible to observe the shift at a level of individual features.

This is how it might look in a binary classification task, like churn prediction. The distribution of a specific feature is stable. But the share of the target class at a particular value range grows over time. A new predictive pattern appears. But this is just a single feature: the initial effect on the model performance is mild.



A change in the relationship between a given feature and the prediction target. (Image by author).

How fast do models age?

It depends, of course. But it is often possible to get an estimate ahead of time.

If we are building a supervised prediction model, we know the past ground truth. We can train the model on older data and apply it to later periods. Then we can imitate model retraining with different frequencies and measure how it impacts model quality.

Such a test gives us a good proxy on regular model retraining needs. Does each new week's data improve the performance? Or a 3-month old model still performs as good as new?

We can then schedule model updates on freshly accumulated data at a chosen interval.

We describe more checks one can run to decide on the retraining schedule in a separate blog.

Sudden concept drift

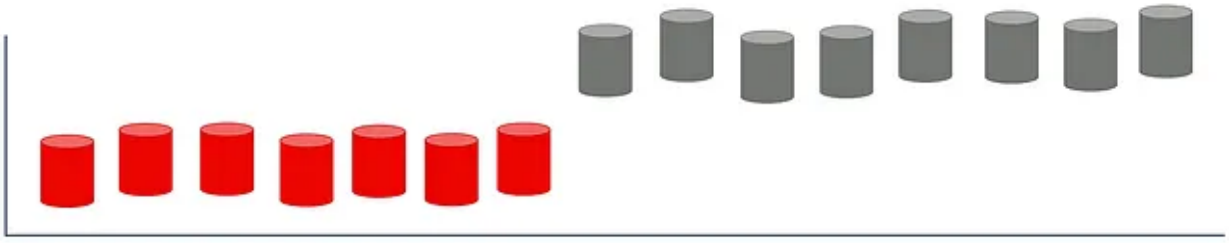


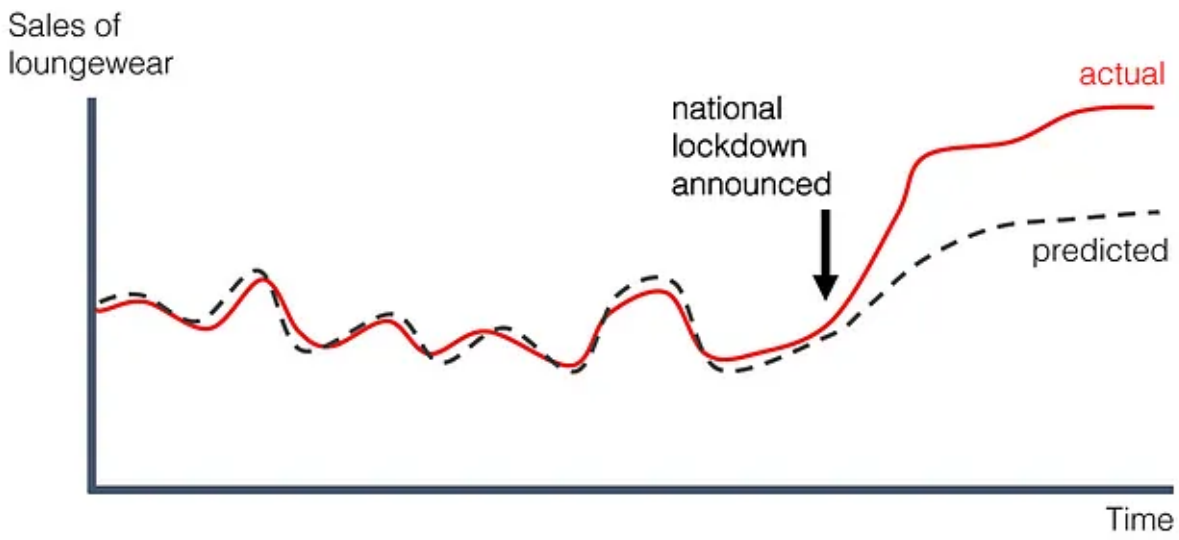
Image by author.

External changes might be more **sudden** or **drastic**. These are hard to miss. As we shared in our recent post, the COVID-19 pandemic is a perfect example.

Almost overnight, the mobility and shopping patterns shifted. It affected all sorts of models, even otherwise "stable" ones.

Demand forecasting models would not guess that sales of yoga pants surge 350% (like it happened to Stitch Fix) or that most flights will be canceled as borders close.

Were you working to recognize pneumonia on X-ray images? You instantly got a new label.



Consumer demand changes suddenly due to stay-at-home policy. (Image by author).

Such abrupt changes do not always need a pandemic or a stock market crash.

In a more ordinary course of events, you can experience things like:

- **Change in the interest rate by the central bank.** All financial and investment behavior is affected, and models fail to adapt to unseen patterns.
- **Technical revamp of the production line.** Predictive maintenance becomes obsolete since modified equipment has new failure modes (or lack of those).
- **Major update in the app interface.** Past data on clicks and conversion becomes irrelevant since the user journey is a new one.

Dealing with drift

Image credit: Unsplash.

In the case of radical shifts, models break. In the meantime, the not-so-relevant model might need human help. You might want to pause it until more data is collected. As a fallback strategy, you can use expert rules or heuristics.

To get back on track, we need to retrain the model. Approaches vary:

- Retrain the model using all available data, both before and after the change.
- Use everything, but assign higher weights to the new data so that model gives priority to the recent patterns.
- If enough new data is collected, we can simply drop the past.

"Naive" retraining is not always enough. If the problem has evolved, we might need to tune the model, not feed the latest data into the existing one.

Options include domain adaptation strategies, building a composition of models that use both old and new data, adding new data sources, or trying entirely new architectures.

Sometimes it's best to modify the model scope or the business process. It can be shortening the prediction horizon or running the model more frequently. For example, retailers facing shifts in shopping habits might switch from weekly to daily demand forecast.

Communication matters, too. Like with interface updates, changes can come from inside the business. Here, you need to align business owners and model maintainers. An early warning will help you prepare the model for a new cold start.

Recurring concept drift

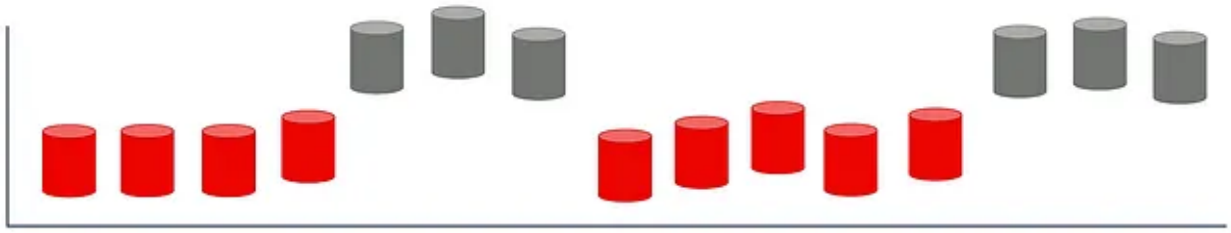


Image by author.

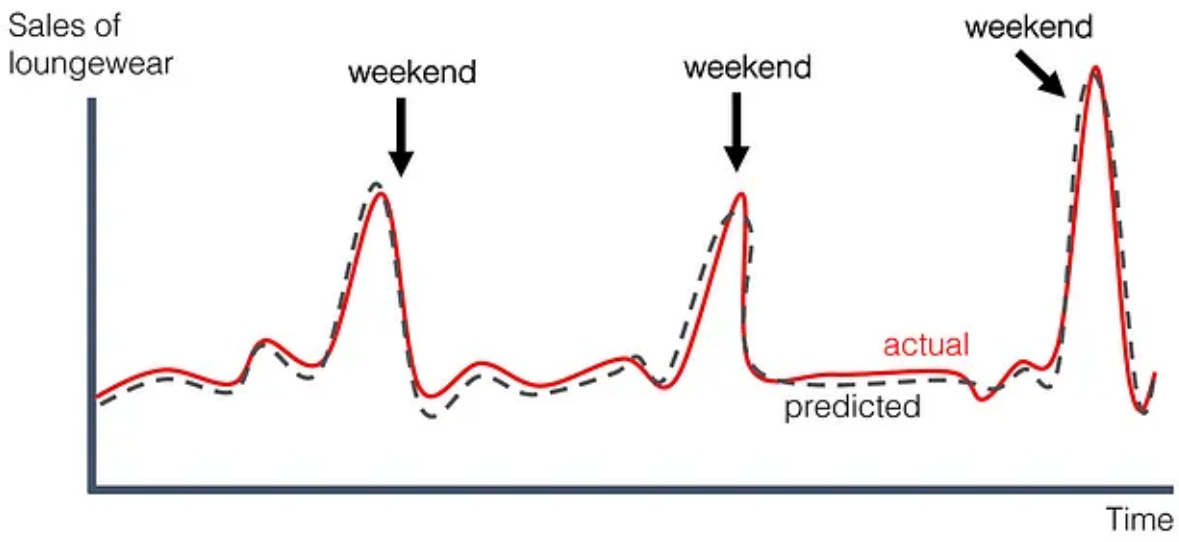
One more honorable mention.

Some researchers use the term "**recurring** drift" to describe repeating changes. We feel here things get mixed up a bit.

Seasonality is a known modeling concept. It indeed looks like (and it is) a temporary change in the target function.

The same people that shop modestly during the year show unusual patterns on Black Fridays. Bank holidays affect everything, from retail sales to production failures. Mobility on the weekend is different from business days. And so on.

If we have a sound model, it should react to these patterns. Every Black Friday resembles the one a year before. We can factor cyclic changes and special events in the system design or build ensemble models. "Recurring drift" will be expected and will not lead to quality decline.



Consumers shop more on the weekends. The pattern is known, and the model forecast shows it. (Image by author).

From the point of model monitoring, this "drift" has no importance. Weekends happen every week, and we don't need an alert. Unless we see a new pattern, of course.

How to treat this not-exactly-drift in production?

Teach your model the seasonality.

If it first sees some special events or seasons in production, you can use other similar events as examples. For instance, if a new bank holiday is introduced, you can assume a similarity with a known one.

If needed, domain experts can help to add manual post-processing rules or corrective coefficients on top of the model output.

Drift recap

Let's sum up.

All models degrade. Sometimes, the performance drop is due to low data quality, broken pipelines, or technical bugs.

If not, you have two more bets:

1. Data drift: change in data distributions.
The model performs worse on unknown data regions.

2. Concept drift: change in relationships.
The world has changed, and the model needs an update. It can be gradual (expected), sudden (you get it), and recurring (seasonal).

In practice, the semantic distinction makes little difference. More often than not, the drift will be combined and subtle.

What matters is how it impacts the model performance, if retraining is justified, and how to catch this on time.

How to analyze this? Explore the tutorial "How to break a model in 20 days" for an applied example and check out our open-source tools for model analytics on GitHub.

This blog was originally published at https://evidentlyai.com. Thanks to my co-founder Emeli Dral for co-authoring the article.

At Evidently AI, we build open-source tools to analyze and monitor machine learning models.

Want to stay in the loop? Sign up for our updates and product news, follow us on Twitter and Linkedin for more content on production machine learning, or join our Discord community to chat and connect.

Machine Learning | Data Science | Artificial Intelligence | Dáta | Mlops

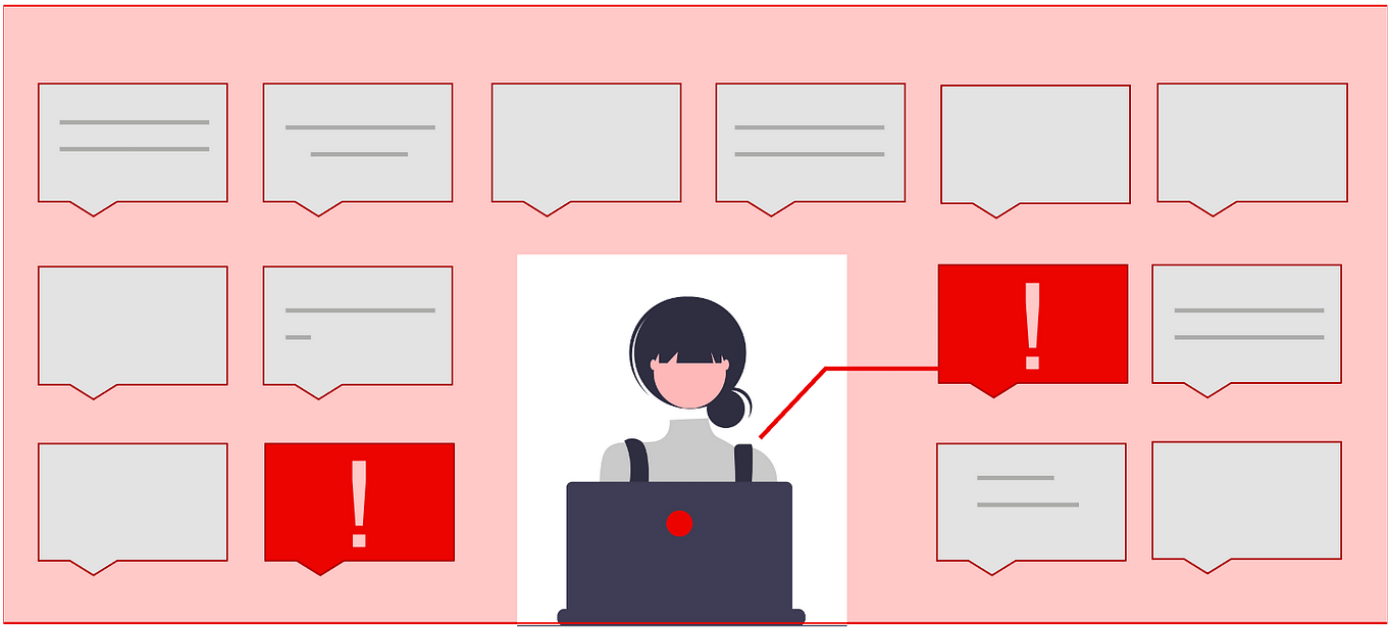
Follow

Written by Elena Samuylova

876 Followers · Writer for Towards Data Science

Co-founder and CEO Evidently AI. Building open-source tools to analyze and monitor machine learning models.

More from Elena Samuylova and Towards Data Science



Elena Samuylova in Towards Data Science

Monitoring NLP models in production

A code tutorial on detecting drift in text data

13 min read · Feb 20

327 5

Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

3.1K 39

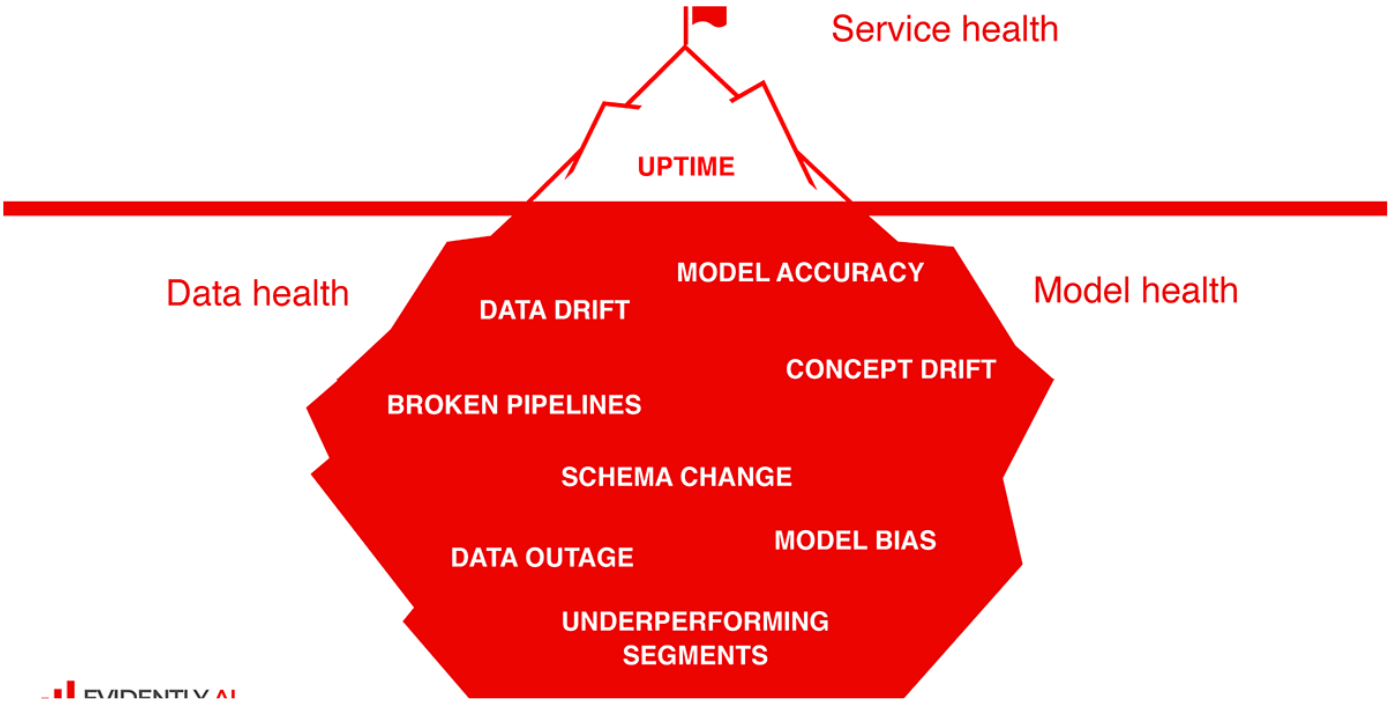
Leonie Monigatti in Towards Data Science

Getting Started with LangChain: A Beginner's Guide to Building LLM-Powered Applications

A LangChain tutorial to build anything with large language models in Python

· 12 min read · Apr 25

2.2K 18



Elena Samuylova in Towards Data Science

Monitoring ML systems in production—which metrics should you track?

When one mentions “ML monitoring,” this can mean many things. Are you tracking service latency? Model accuracy? Data quality? The share of...

24 min read · Jul 18, 2022

33

See all from Elena Samuylova

See all from Towards Data Science

Recommended from Medium

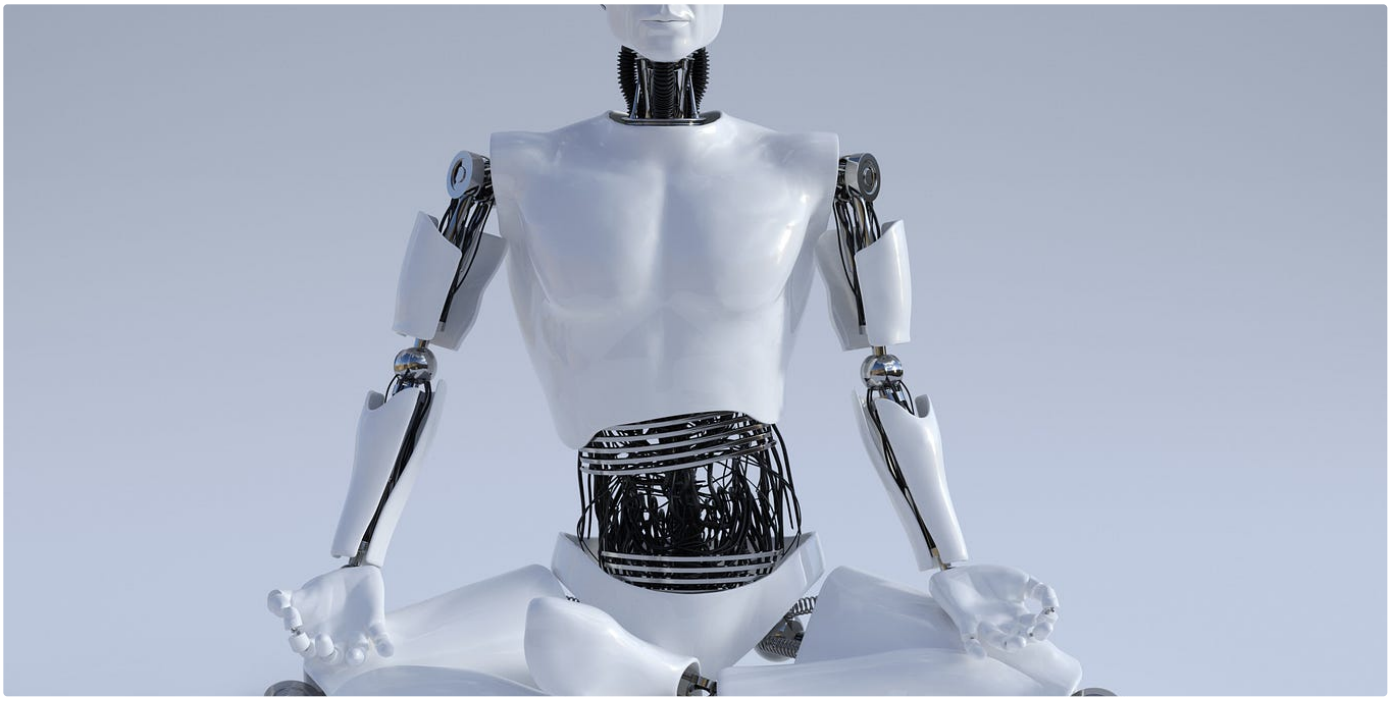
 Leonie Monigatti in Towards Data Science

Getting Started with LangChain: A Beginner's Guide to Building LLM-Powered Applications

A LangChain tutorial to build anything with large language models in Python

✨ · 12 min read · Apr 25

 2.2K  18 



The PyCoach in Artificial Corner

You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of ChatGPT Users

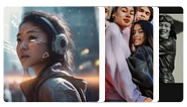
Master ChatGPT by learning prompt engineering.

· 7 min read · Mar 17

Lists

What is ChatGPT?

9 stories · 61 saves



Staff Picks

329 stories · 82 saves

Stories to Help You Level-Up at Work

19 stories · 53 saves

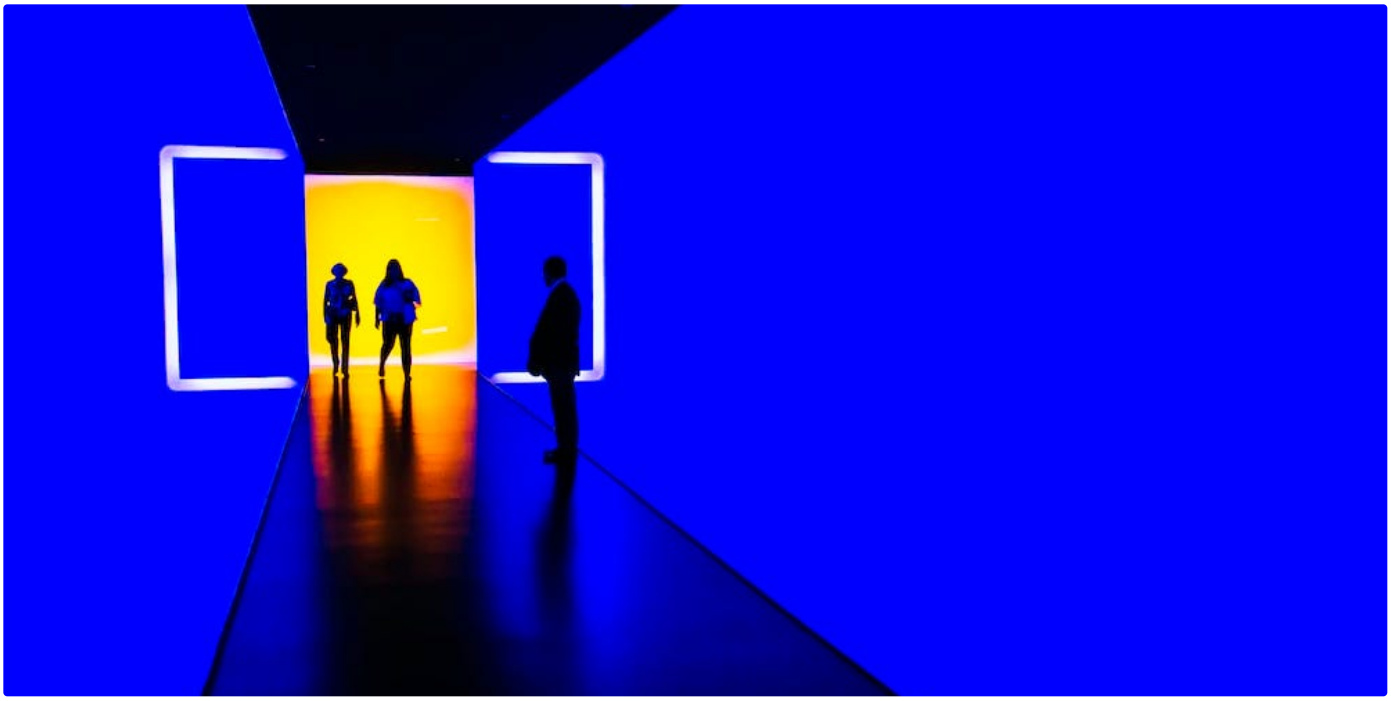
Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews...

They don't meet the bar.

· 4 min read · Apr 13

4.2K 127



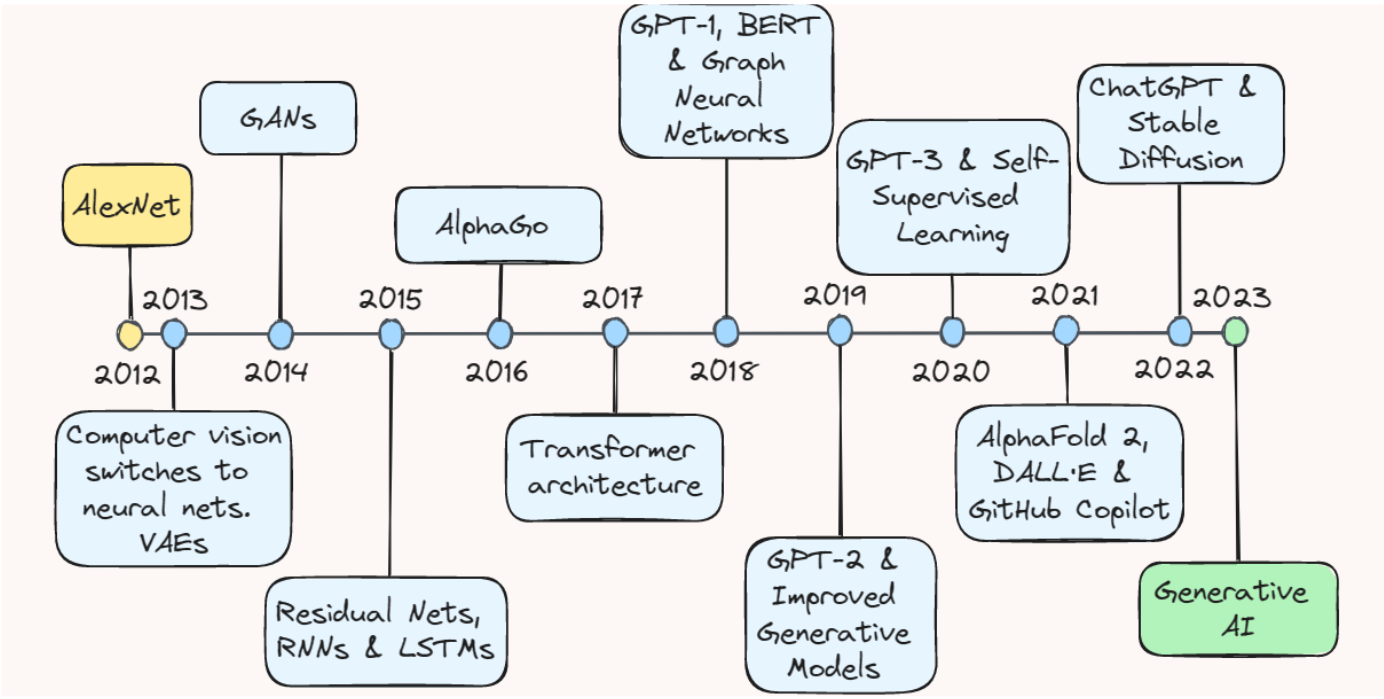
Matt Chapman in Towards Data Science

How I Stay Up to Date With the Latest AI Trends as a Full-Time Data Scientist

No, I don't just ask ChatGPT to tell me

✨ · 8 min read · May 1

1.4K 21



Thomas A Dorfer in Towards Data Science

Ten Years of AI in Review

From image classification to chatbot therapy

✨ · 15 min read · May 23

912 9

 Matt Chapman in Towards Data Science

The Portfolio that Got Me a Data Scientist Job

Spoiler alert: It was surprisingly easy (and free) to make

✨ · 10 min read · Mar 24

2.9K 44

See more recommendations