



Machine Learning in Production

Testing in Production

Back to QA...

Fundamentals of Engineering AI-Enabled Systems

Holistic system view: AI and non-AI components, pipelines, stakeholders, environment interactions, feedback loops

Requirements:

System and model goals
User requirements
Environment assumptions
Quality beyond accuracy
Measurement
Risk analysis
Planning for mistakes

Architecture + design:

Modeling tradeoffs
Deployment architecture
Data science pipelines
Telemetry, monitoring
Anticipating evolution
Big data processing
Human-AI design

Quality assurance:

Model testing
Data quality
QA automation
Testing in production
Infrastructure quality
Debugging

Operations:

Continuous deployment
Contin. experimentation
Configuration mgmt.
Monitoring
Versioning
Big data
DevOps, MLOps

Teams and process: Data science vs software eng. workflows, interdisciplinary teams, collaboration points, technical debt

Responsible AI Engineering

Provenance,
versioning,
reproducibility

Safety

Security and
privacy

Fairness

Interpretability
and explainability

Transparency
and trust

Ethics, governance, regulation, compliance, organizational culture

Learning Goals

- Design telemetry for evaluation in practice
- Understand the rationale for beta tests and chaos experiments
- Plan and execute experiments (chaos, A/B, shadow releases, ...) in production
- Conduct and evaluate multiple concurrent A/B tests in a system
- Perform canary releases
- Examine experimental results with statistical rigor
- Support data scientists with monitoring platforms providing insights from production data

Readings

Required Reading:

- Hulten, Geoff. "[Building Intelligent Systems: A Guide to Machine Learning Engineering.](#)" Apress, 2018, Chapters 14 and 15 (Intelligence Management and Intelligent Telemetry).

Suggested Readings:

- Alec Warner and Štěpán Davidovič. "[Canary Releases.](#)" in [The Site Reliability Workbook](#), O'Reilly 2018
- Kohavi, Ron, Diane Tang, and Ya Xu. "[Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing.](#)" Cambridge University Press, 2020.

From Unit Tests to Testing in Production

(in traditional software systems)

Unit Test, Integration Tests, System Tests



Unit testing

Integration testing

System testing

Acceptance
testing
(Demonstration)

Speaker notes

Testing before release. Manual or automated.



Beta Testing



Speaker notes

Early release to select users, asking them to send feedback or report issues. No telemetry in early days.



Crash Telemetry

Crash2.exe

Crash2.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

Speaker notes

With internet availability, send crash reports home to identify problems "in production". Most ML-based systems are online in some form and allow telemetry.



A/B Testing

Original: 2.3%

The original landing page for Groove. It features a large image of a smiling man in a plaid shirt. Text on the left reads "SaaS & eCommerce Customer Support." and a quote from "Griffin, Customer Champion at Allocacoo": "Managing customer support requests in Groove is so easy. Way better than trying to use Gmail or a more complicated help desk." Below the quote is a testimonial: "97% of customers recommend Groove." A green "Learn More" button is at the bottom right.

Long Form: 4.3%

The long-form landing page for Groove. It has a larger headline: "Everything you need to deliver awesome, personal support to every customer." Below it is a sub-headline: "Assign support emails to the right people, feel confident that customers are being followed up with and always know what's going on." A video player shows a man named Allan using Groove. To the right is a sidebar with "WHAT YOU'LL DISCOVER ON THIS PAGE" and a list of bullet points: "Three reasons growing teams choose Groove", "How Groove makes your whole team more productive", "Delivering a personal support experience every time", "Take a personalized tour", and "A personal note from our CEO". Logos for BuySellAds, METALAB, and StatusPage.io are at the bottom.

Speaker notes

Usage observable online, telemetry allows testing in production. Picture source: <https://www.designforfounders.com/ab-testing-examples/>



Chaos Experiments



Speaker notes

Deliberate introduction of faults in production to test robustness.



Model Assessment in Production

Ultimate held-out evaluation data: Unseen real user data

Limitations of Offline Model Evaluation

Training and test data drawn from the same population

- **i.i.d.: independent and identically distributed**
- leakage and overfitting problems quite common

Is the population representative of production data?

If not or only partially or not anymore: Does the model generalize beyond training data?

Identify Feedback Mechanism in Production

Live observation in the running system

Potentially on subpopulation (A/B testing)

Need telemetry to evaluate quality -- challenges:

- Gather feedback without being intrusive (i.e., labeling outcomes), without harming user experience
- Manage amount of data
- Isolating feedback for specific ML component + version

Discuss how to collect feedback

More:

- SmartHome: Does it automatically turn off the lights/lock the doors/close the window at the right time?
- Profanity filter: Does it block the right blog comments?
- News website: Does it pick the headline alternative that attracts a user's attention most?
- Autonomous vehicles: Does it detect pedestrians in the street?





Speaker notes

Expect only sparse feedback and expect negative feedback over-proportionally





DFW ↔ SFO

1659 of 1687 flights

Nov 16

Wednesday

Advice: **Watch** Learn more ⓘ

Create a

Stops

nonstop

1 stop

2+ stops

Prices may fall within 7 days – Watch

Our model strongly indicates that fares will fall during the next 7 days. This forecast is based on analysis of historical price changes and is not a guarantee of future results.

Times

Create a price alert

Take-off Dallas

Speaker notes

Can just wait 7 days to see actual outcome for all predictions





00:00  Offset 00:00 01:31:27

▶ Play ⏪ Back 5s 1x Speed 🔊 Volume

NOTES

Write your notes here

Speaker 5 ▶ 07:44

Yeah. So there's a slight story behind that. So back when I was in, uh, Undergrad, I wrote a program for myself to measure a, the amount of time I did data entry from my father's business and I was on windows at the time and there wasn't a function called time dot [inaudible] time, uh, which I needed to parse dates to get back to time, top of representation, uh, I figured out a way to do it and I gave it to what's called the python cookbook because it just seemed like something other people could use. So it was just trying to be helpful. Uh, subsequently I had to figure out how to make it work because I didn't really have to. Basically, it bothered me that you had to input all the locale information and I figured out how to do it over the subsequent months. And actually as a graduation gift from my Undergrad, the week following, I solved it and wrote it all out.

Speaker 5 ▶ 08:38

And I asked, uh, Alex Martelli, the editor of the Python Cookbook, which had published my original recipe, a, how do I get this into python? I think it might help

How did we do on your transcript? 

Speaker notes

Clever UI design allows users to edit transcripts. UI already highlights low-confidence words, can



Manually Label Production Samples

Similar to labeling learning and testing data, have human annotators



Summary: Telemetry Strategies

- Wait and see
- Ask users
- Manual/crowd-source labeling, shadow execution
- Allow users to complain
- Observe user reaction

Breakout: Design Telemetry in Production

Discuss how to collect telemetry (Wait and see, ask users, manual/crowd-source labeling, shadow execution, allow users to complain, observe user reaction)

Scenarios:

- Front-left: Amazon: Shopping app feature that detects the shoe brand from photos
- Front-right: Google: Tagging uploaded photos with friends' names
- Back-left: Spotify: Recommended personalized playlists
- Back-right: Wordpress: Profanity filter to moderate blog posts

≡ (no need to post in slack yet)

Measuring Model Quality with Telemetry

- Usual 3 steps: (1) Metric, (2) data collection (telemetry), (3) operationalization
- Telemetry can provide insights for correctness
 - sometimes very accurate labels for real unseen data
 - sometimes only mistakes
 - sometimes delayed
 - often just samples
 - often just weak proxies for correctness
- Often sufficient to *approximate* precision/recall or other model-quality measures
- Mismatch to (static) evaluation set may indicate stale or unrepresentative data
- Trend analysis can provide insights even for inaccurate proxy measures

Breakout: Design Telemetry in Production

Discuss how to collect telemetry, the metric to monitor, and how to operationalize

Scenarios:

- Front-left: Amazon: Shopping app detects the shoe brand from photos
- Front-right: Google: Tagging uploaded photos with friends' names
- Back-left: Spotify: Recommended personalized playlists
- Back-right: Wordpress: Profanity filter to moderate blog posts

As a group post to #lecture and tag team members:

Speaker notes

about 30 minutes to here



Monitoring Model Quality in Production

- Monitor model quality together with other quality attributes (e.g., uptime, response time, load)
- Set up automatic alerts when model quality drops
- Watch for jumps after releases
 - roll back after negative jump
- Watch for slow degradation
 - Stale models, data drift, feedback loops, adversaries
- Debug common or important problems
 - Monitor characteristics of requests
 - Mistakes uniform across populations?
 - Challenging problems -> refine training, add regression tests

Average rating last 15min



Prometheus and Grafana





Many commercial solutions



e.g. <https://www.datarobot.com/platform/mlops/>



Many pointers: Ori Cohen "Monitor! Stop Being A Blind Data-Scientist." Blog 2019

Detecting Drift



Image source: Joel Thomas and Clemens Mewald. [Productionizing Machine Learning: From Deployment to Drift Detection](#). Databricks Blog, 2019

Engineering Challenges for Telemetry

Engineering Challenges for Telemetry

- Data volume and operating cost
 - e.g., record "all AR live translations"?
 - reduce data through sampling
 - reduce data through summarization (e.g., extracted features rather than raw data; extraction client vs server side)
- Adaptive targeting
- Biased sampling
- Rare events
- Privacy
- Offline deployments?

Breakout: Engineering Challenges in Telemetry

Discuss: Cost, privacy, rare events, bias

Scenarios:

- Front-left: Amazon: Shopping app feature that detects the shoe brand from photos
- Front-right: Google: Tagging uploaded photos with friends' names
- Back-left: Spotify: Recommended personalized playlists
- Back-right: Wordpress: Profanity filter to moderate blog posts

(can update slack, but not needed)



Telemetry for Training: The ML Flywheel



≡ graphic by [CBInsights](#)

Revisiting Model Quality vs System Goals

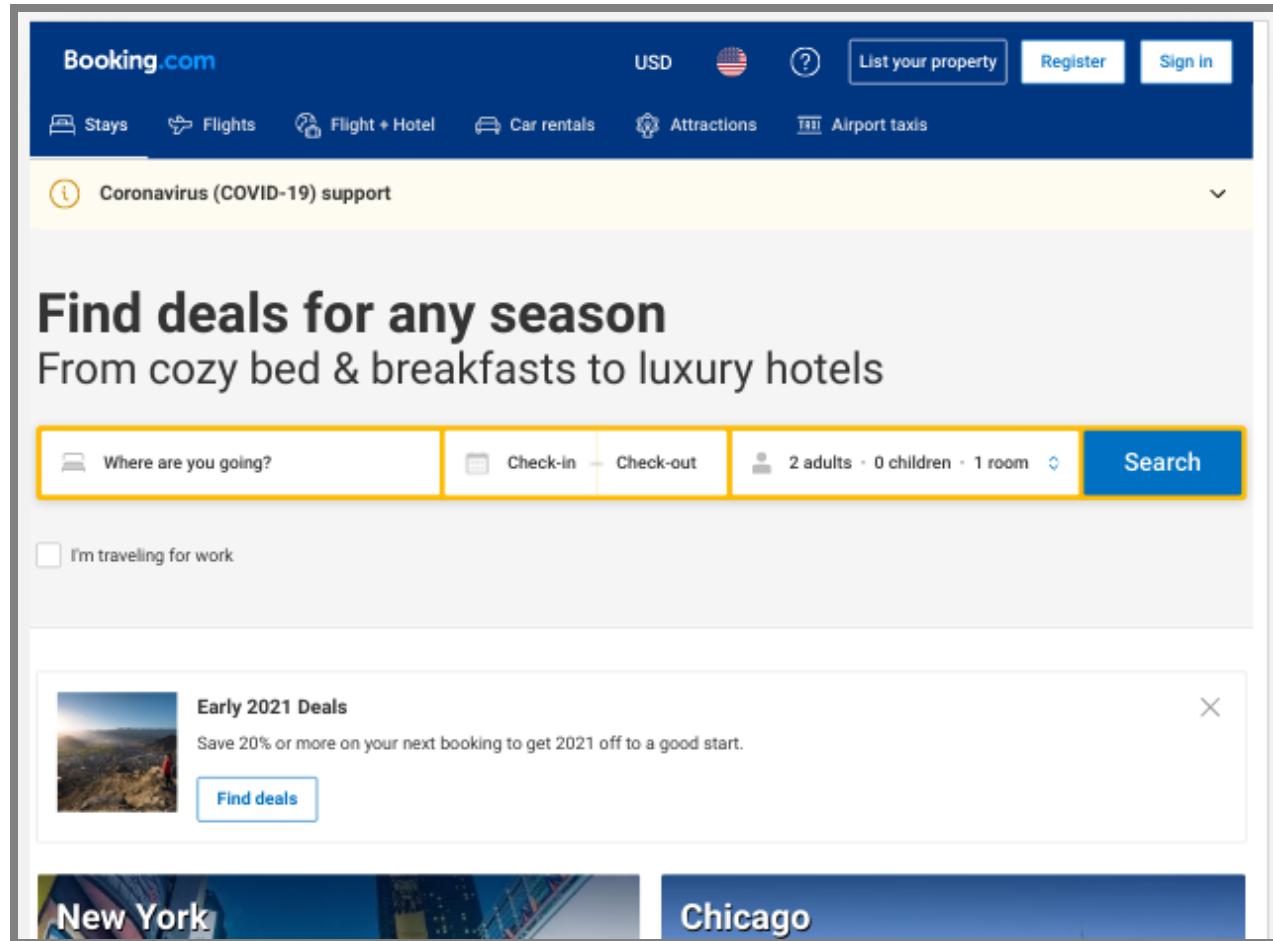
Model Quality vs System Goals

Telemetry can approximate model accuracy

Telemetry can directly measure system qualities, leading indicators, user outcomes

- define measures for "key performance indicators"
- clicks, buys, signups, engagement time, ratings
- operationalize with telemetry

Model Quality vs System Quality



Bernardi, Lucas, et al. "150 successful machine learning models: 6 lessons learned at Booking.com." In Proc. Int'l Conf. Knowledge Discovery & Data Mining, 2019.

Possible causes of model vs system conflict?



hypothesized

- model value saturated, little more value to be expected
- segment saturation: only very few users benefit from further improvement
- overoptimization on proxy metrics not real target metrics
- uncanny valley effect from "creepy AIs"

Breakout: Design Telemetry in Production

Discuss: What key performance indicator of the system to collect?

Scenarios:

- Front-left: Amazon: Shopping app feature that detects the shoe brand from photos
- Front-right: Google: Tagging uploaded photos with friends' names
- Back-left: Spotify: Recommended personalized playlists
- Back-right: Wordpress: Profanity filter to moderate blog posts

(can update slack, but not needed)

Experimenting in Production

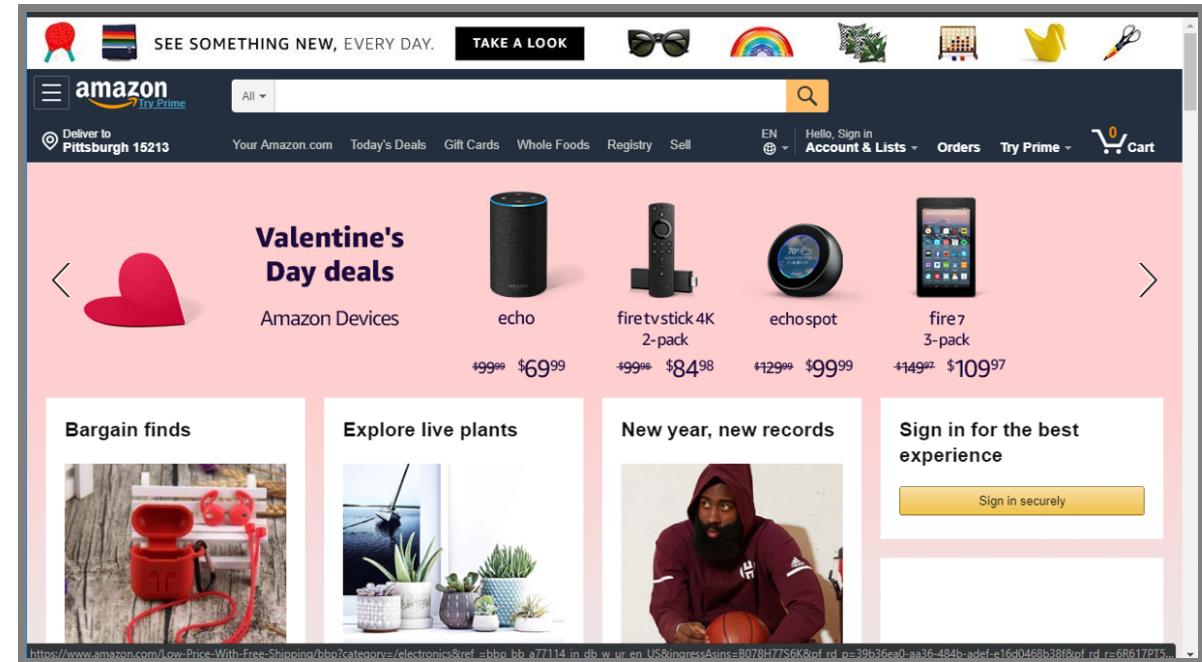
- A/B experiments
- Shadow releases / traffic teeing
- Blue/green deployment
- Canary releases
- Chaos experiments

A/B Experiments

What if...?

- ... we hand plenty of subjects for experiments
- ... we could randomly assign to treatment and control group without them knowing
- ... we could analyze small individual changes and keep everything else constant

► Ideal conditions for controlled experiments



A/B Testing for Usability

- In running system, random users are shown modified version
- Outcomes (e.g., sales, time on site) compared among groups

Original: 2.3%

The original version of the Groove website features a large image of a smiling man in a plaid shirt. The headline reads "SaaS & eCommerce Customer Support." Below it, a quote from Griffin, a Customer Champion at Allocate, says, "Managing customer support requests in Groove is so easy. Way better than trying to use Gmail or a more complicated help desk." A testimonial bar shows "97% of customers recommend Groove." A green "Learn More" button is prominent. The footer includes links for "How it works," "What you get," "What it costs," and "How we're different." A bold statement at the bottom says, "You'll be up and running in less than a minute."

Long Form: 4.3%

The long-form version of the Groove website adds a "Sign Up" button to the top right. The main headline is "Everything you need to deliver awesome, personal support to every customer." It includes a video player showing a man named Allan speaking. A sidebar lists "WHAT YOU'LL DISCOVER ON THIS PAGE" with items like "Three reasons growing teams choose Groove" and "How Groove makes your whole team more productive." Logos for "1500+ HAPPY CUSTOMERS" and various partners like BuySellAds, METALAB, and StatusPage.io are at the bottom.

Speaker notes

Picture source: <https://www.designforfounders.com/ab-testing-examples/>



Bing Experiment



- Experiment: Ad Display at Bing
- Suggestion prioritized low
- Not implemented for 6 month
- Ran A/B test in production
- Within 2h *revenue-too-high* alarm triggered suggesting serious bug (e.g., double billing)
- Revenue increase by 12% - \$100M annually in US
- Did not hurt user-experience metrics

From: Kohavi, Ron, Diane Tang, and Ya Xu.
"Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing." 2020.

A/B Experiment for ML Components?

- New product recommendation algorithm for web store?
- New language model in audio transcription service?
- New (offline) model to detect falls on smart watch



Experiment Size

With enough subjects (users), we can run many experiments

Even very small experiments become feasible

Toward causal inference

Get Started Now
It's free! No trials, no fees.

vs

Get Started Now
It's free! No trials, no fees.

Speaker notes

about an hour in



Implementing A/B Testing

Implement alternative versions of the system

- using feature flags (decisions in implementation)
- separate deployments (decision in router/load balancer)

Map users to treatment group

- Randomly from distribution
- Static user - group mapping
- Online service (e.g., [launchdarkly](#), [split](#))

Monitor outcomes *per group*

- Telemetry, sales, time on site, server load, crash rate

Speaker notes

divide them into groups



Feature Flags (Boolean flags)

```
if (features.enabled(userId, "one_click_checkout")) {  
    // new one click checkout function  
} else {  
    // old checkout functionality  
}
```

- Good practices: tracked explicitly, documented, keep them localized and independent
- External mapping of flags to customers, who should see what configuration
 - e.g., 1% of users sees `one_click_checkout`, but always the same users; or 50% of beta-users and 90% of developers and 0.1% of all users

```
def isEnabled(user): Boolean = (hash(user.id) % 100) < 10
```

Speaker notes

mapping somewhere One way of doing this randomly is hashing user ID random but stable same users always in the same group some offset to get a new sample for telemetry you need to know what group a user was in. Once you have a mapping from flags here then it's easier you can also use a load balancer to manage this you need an if statement somewhere chrome, facebook, if statements are in the backend/code



Treatments ? | 2 treatments, if Split is killed serve the default treatment of "off"

Treatment	Default	Description
on		The new version of registration process is enabled.
off		The old version of registration process is enabled.

[+ Add treatment](#) | [Learn more about multivariate treatments.](#)

Whitelist ? | 0 user(s) or segments individually targeted.

[+ Add whitelist](#)

Traffic Allocation ? | 100% of user included in Split rules evaluation below.

Total Traffic Allocation: 100 % total User in Split

Targeting Rules ? | 2 rules created for targeting.

```
graph TD; A((if)) --> B((user is in segment qa)); B --> C((Then serve on)); A --> D((else if)); D --> E((user is in segment beta_testers)); E --> F((Then serve percentage)); F --> G((on 50)); F --> H((off 50));
```

[+ Add rule](#)

Default Rule ? | Serve treatment of "off".

serve off



Speaker notes

There are companies that do this as a service exposre your database. "boolean option as a service" most groups in the past have done it themselves

launch darkley, pslit IO, open source libraries gets complicated when you want to do multiple experiments, factorial design gets complicated the bing team wrote a whole book



Confidence in A/B Experiments

(statistical tests)

Comparing Averages

Group A

*classic personalized content
recommendation model*

2158 Users

average 3:13 min time on site

Group B

*updated personalized content
recommendation model*

10 Users

average 3:24 min time on site

Speaker notes

What's the problem here? t test assumes a normal distribution

statistical tests to see if it's random event

who here knows about the t test? It's one of the standard tests for this kind of thing.



Comparing Distributions



Speaker notes

means of the real distributions from which these are sampled, chances that they're actually different

t test captures that, it's old/simple, it's robust enough for almost everything we're doing here

Originally for agriculture, all about barley vs. hops, it's about beer

milestone 3 we're going to ask for statistical confidence, T test is fine.

all these tools implement this .



Different effect size, same deviations



Same effect size, different deviations



Less noise --> Easier to recognize

Dependent vs. independent measurements

Pairwise (dependent) measurements

- Before/after comparison
- With same benchmark + environment
- e.g., new operating system/disc drive faster

Independent measurements

- Repeated measurements
- Input data regenerated for each measurement

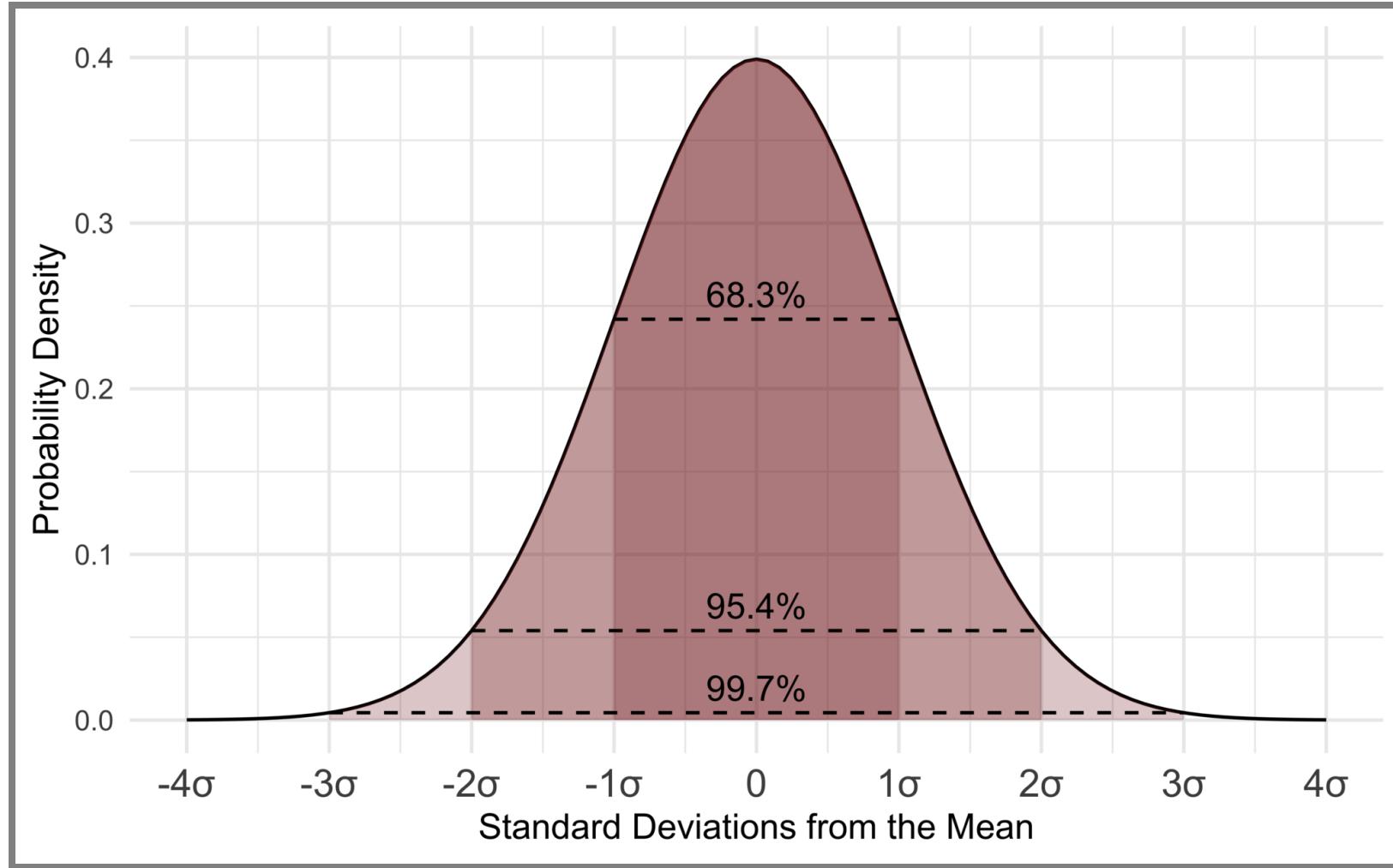
Significance level

- Statistical change of an error
- Define before executing the experiment
 - use commonly accepted values
 - based on cost of a wrong decision
- Common:
 - 0.05 significant
 - 0.01 very significant
- Statistically significant result \Rightarrow proof
- Statistically significant result \Rightarrow important result
- Covers only alpha error (more later)

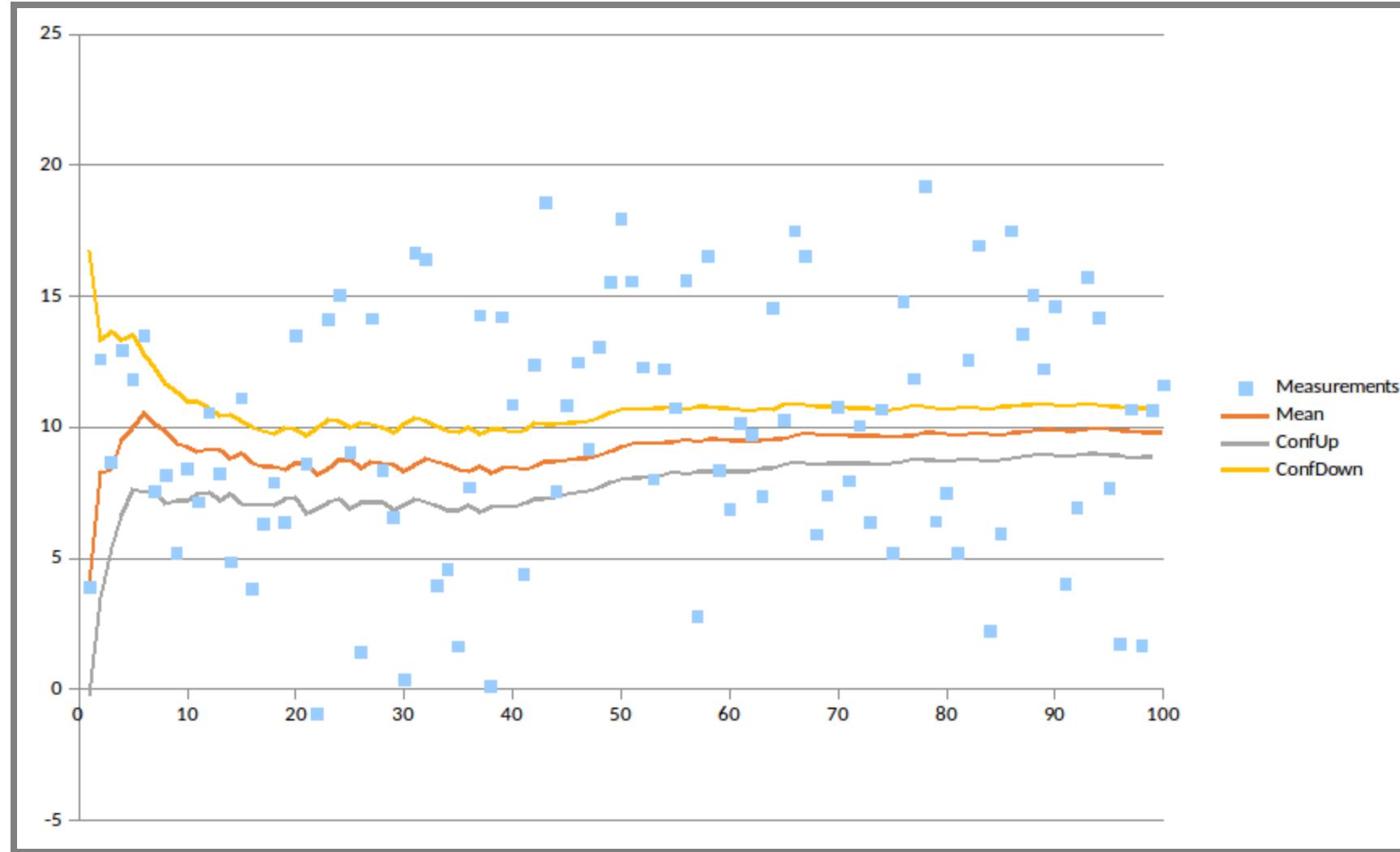
Intuition: Error Model

- 1 random error, influence +/- 1
 - Real mean: 10
 - Measurements: 9 (50%) und 11 (50%)
-
- 2 random errors, each +/- 1
 - Measurements: 8 (25%), 10 (50%) und 12 (25%)
-
- 3 random errors, each +/- 1
 - Measurements : 7 (12.5%), 9 (37.5), 11 (37.5), 12 (12.5)

Normal Distribution



Confidence Intervals



Comparison with Confidence Intervals



Source: Andy Georges, et al. 2007. [Statistically rigorous java performance evaluation](#). In Proc. Conference on Object-Oriented Programming Systems and Applications.

t-test

```
> t.test(x, y, conf.level=0.9)
```

Welch Two Sample t-test

t = 1.9988, df = 95.801, p-value = 0.04846

alternative hypothesis: true difference in means is
not equal to 0

90 percent confidence interval:

0.3464147 3.7520619

sample estimates:

mean of x mean of y



Source: <https://conversionsciences.com/ab-testing-statistics/>



Source: <https://cognetik.com/why-you-should-build-an-ab-test-dashboard/>

How many samples needed?

Too few?

Too many?

Speaker notes

50/50? New model vs. old model about 1:10 to here inherently risky, so you don't want too many in experiment



A/B testing automation

- Experiment configuration through DSLs/scripts
- Queue experiments
- Stop experiments when confident in results
- Stop experiments resulting in bad outcomes (crashes, very low sales)
- Automated reporting, dashboards

Further readings:

DSL for scripting A/B tests at Facebook

```
button_color = uniformChoice(  
    choices=['#3c539a', '#5f9647', '#b33316'],  
    unit=cookieid);  
  
button_text = weightedChoice(  
    choices=['Sign up', 'Join now'],  
    weights=[0.8, 0.2],  
    unit=cookieid);  
  
if (country == 'US') {  
    has_translate = bernoulliTrial(p=0.2, unit=userid);
```

Further readings: Bakshy, Eytan et al. [Designing and deploying online field experiments](#). Proc.  WWW, 2014. (Facebook)

Concurrent A/B testing

Multiple experiments at the same time

- Independent experiments on different populations -- interactions not explored
- Multi-factorial designs, well understood but typically too complex, e.g., not all combinations valid or interesting
- Grouping in sets of experiments (layers)

Further readings:

Other Experiments in Production

Shadow releases / traffic teeing

Blue/green deployment

Canary releases

Chaos experiments

Shadow releases / traffic teeing

Run both models in parallel

Use predictions of old model in production

Compare differences between model predictions

If possible, compare against ground truth labels/telemetry

Examples?

Blue/green deployment

Provision service both with old and new model (e.g., services)

Support immediate switch with load-balancer

Allows to undo release rapidly

Advantages/disadvantages?

Canary Releases

Release new version to small percentage of population (like A/B testing)

Automatically roll back if quality measures degrade

Automatically and incrementally increase deployment to 100% otherwise



Chaos Experiments



Chaos Experiments for ML Components?



Speaker notes

Artifically reduce model quality, add delays, insert bias, etc to test monitoring and alerting infrastructure



Advice for Experimenting in Production

Minimize *blast radius* (canary, A/B, chaos expr)

Automate experiments and deployments

Allow for quick rollback of poor models (continuous delivery, containers, loadbalancers, versioning)

Make decisions with confidence, compare distributions

Monitor, monitor, monitor

Bonus: Monitoring without Ground Truth

Invariants/Assertions to Assure with Telemetry

- Consistency between multiple sources
 - e.g., multiple models agree, multiple sensors agree
 - e.g., text and image agree
- Physical domain knowledge
 - e.g., cars in video shall not flicker,
 - e.g., earthquakes should appear in sensors grouped by geography
- Domain knowledge about unlikely events
 - e.g., unlikely to have 3 cars in same location
- Stability
 - e.g., object detection should not change with video noise
- Input conforms to schema (e.g. boolean features)
- And all invariants from model quality lecture, including capabilities

Summary

Production data is ultimate unseen validation data

Both for model quality and system quality

Telemetry is key and challenging (design problem and opportunity)

Monitoring and dashboards

Many forms of experimentation and release (A/B testing, shadow releases, canary releases, chaos experiments, ...) to minimize "blast radius"; gain confidence in results with statistical tests

Further Readings

- On canary releases: Alec Warner and Štěpán Davidovič. “[Canary Releases.](#)” in [The Site Reliability Workbook](#), O’Reilly 2018
- Everything on A/B testing: Kohavi, Ron. [*Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*](#). Cambridge University Press, 2020.
- A/B testing critiques: Josh Constine. [The Morality Of A/B Testing](#). Blog 2014; the [Center of Humane Technology](#); and the Netflix documentary [The Social Dilemma](#)
- Ori Cohen “[Monitor! Stop Being A Blind Data-Scientist.](#)” Blog 2019
- Jens Meinicke, Chu-Pan Wong, Bogdan Vasilescu, and Christian Kästner. [Exploring Differences and Commonalities between Feature Flags and Configuration Options](#). In Proceedings of the Proc. International Conference on Software Engineering ICSE-SEIP, pages 233–242, May 2020.

