Data Analysis

Model failures and monitoring

Topics

Natural labels & feedback loops

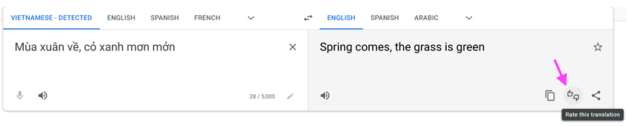
Natural labels

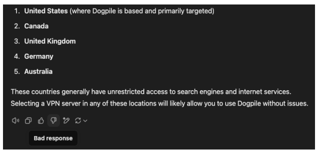
- The model's predictions can be automatically evaluated or partially evaluated by the system.
- Examples:
  - ETA
  - Ride demand prediction
  - Stock price prediction
  - Ads CTR (click-through-ratio)
  - Recommender system

Post Test
↓
In Product

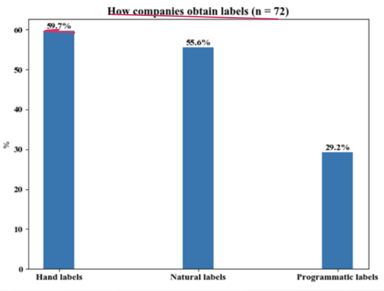
Natural labels

- You can engineer a task to have natural labels





Natural labels: surprisingly common



⚠ Biases ⚠

- Small sample size
- Companies might only use ML for tasks with *natural* labels

Claypot AI's real-time ML survey (2022)

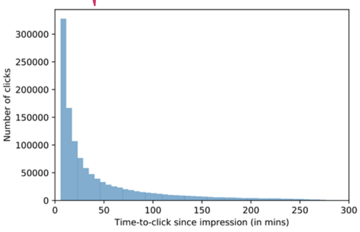
⚠ Labels are often assumed ⚠

We try to infer them

- Recommendation:
  - Click -> good rec
  - After X minutes, no click -> bad rec

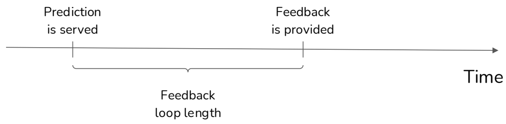
Too short → False negatives

Too long → Slow feedback



Addressing Delayed Feedback for Continuous Training with Neural Networks in CTR prediction (Ktena et al., 2019)

Delayed labels



- Short feedback loop: minutes -> hours
  - Reddit / Twitter / TikTok's recommender systems
- Long feedback loop: weeks -> months
  - Retail website (e.g. amazon) recommender systems
  - Fraud detection

Causes of ML failures

Amazon scraps secret AI recruiting tool that showed bias against women

That is because Amazon's computer models were trained to vet applicants by observing patterns in resumes submitted to the company over a 10-year period. Most came from men, a reflection of male dominance across the tech industry.

In effect, Amazon's system taught itself that male candidates were preferable. It penalized resumes that included the word "women's," as in "women's chess club captain." And it downgraded graduates of two all-women's colleges, according to people familiar with the matter. They did not specify the names of the schools.

Amazon scraps secret AI recruiting tool that showed bias against women (Reuters, 2018)

Japan's Henn na Hotel fires half its robot workforce

"Guests complained their robot room assistants thought snoring sounds were commands and would wake them up repeatedly during the night."

https://www.hotelmanagement.net/tech/japan-s-henn-na-hotel-fires-half-its-robot-workforce

What is an ML failure?

We are in Production Mode

A failure happens when one or more expectations of the system is violated.

Two types of expectations:

- Operational metrics: e.g. average latency, throughput, uptime, ...
- ML metrics: e.g. accuracy, F1, BLEU score, ...

- Traditional software: mostly operational metrics
- ML systems: operational + ML metrics
  - Ops: returns a prediction within 100ms latency on average
  - ML: BLEU score of 55 (out of 100)

Combine both

ML system failures

- If you enter a sentence and get no translation back -> ops failure
- If one translation is incorrect -> ML failure?

- Not necessarily: expected BLEU score < 100
- ML failure if translations are consistently incorrect

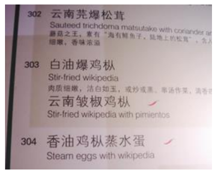
Ops failures

Visible

- 404, timeout, segfault, OOM, etc.

ML failures

Often invisible

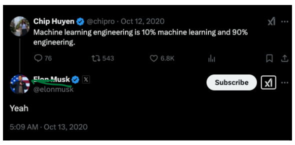


Causes of ops failures (software system failures)

- Dependency failures
- Deployment failures
- ~~Hardware failures~~
- Network failure: downtime / crash

60 / 96 ML systems failures are non-ML failures
(Papasian & Underwood, 2020)

As tooling & best practices around ML production mature, there will be less room for software failures



ML-specific failures (during/post deployment)

1. Production data differing from training data
2. Edge cases
3. Degenerate feedback loops

We talked about some of these already ☺

Production data differing from training data

- Train-serving skew:
  - Model performing well during development but poorly after production
- Data distribution shifts
  - Model performing well when first deployed, but poorly over time
  - ⚠ What looks like data shifts might be caused by human errors ⚠

Common & crucial.
Will go into detail!

Edge cases

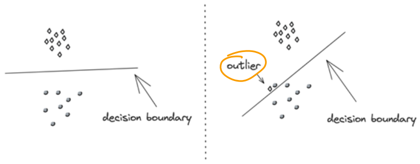
- Self-driving car (yearly)
  - Safely: 99.99%
  - Fatal accidents: 0.01%

Go to wooclap.com
Enter the event code in the top banner
Event code
UMDA
Enable answers by SMS

Poll: Would you use this car?

Edge case vs. outlier

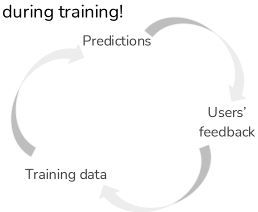
- Outliers
  - Refer to inputs
  - Options to ignore/remove
- Edge cases
  - Refer to outputs
  - Can't ignore/remove



e.g. a person jay-walking on a highway is an outlier, but it's not an edge case if your self-driving car can accurately detect that person and decide on a motion response appropriately.

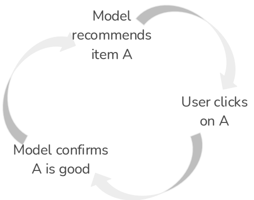
Degenerate feedback loops

- When predictions influence the feedback, which is then used to extract labels to train the next iteration of the model
- Common in tasks with natural labels
- Hard to detect during training!



Degenerate feedback loops: recsys

- Originally, A is ranked marginally higher than B -> model recommends A
- After a while, A is ranked much higher than B



Over time, recommendations become more homogenous

Degenerate feedback loops: resume screening

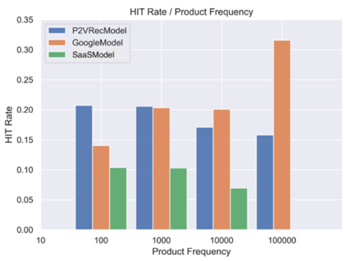
- Originally, model thinks **X** is a good feature
- Model only picks resumes with **X**
- Hiring managers only see resumes with **X**, so only people with **X** are hired
- Model confirms that **X** is good

Replace **X** with e.g.

- Has a name that is typically used for males
- Took the DACS Data Analysis course
- Worked at Google

Degenerate feedback loops: detect

- Average Rec Popularity (ARP)
  - Average popularity of the recommended items
- Average Percentage of Long Tail Items (APLT)
  - average % of long tail items being recommended
- Hit rate against popularity
  - Accuracy based on recommended items' popularity buckets

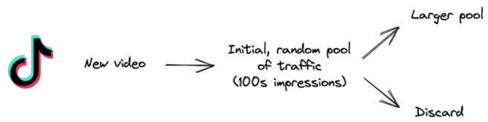


[Beyond NDCG: behavioral testing of recommender systems with RecList](#) (Chia et al., 2021)

Degenerate feedback loops: mitigate

One way is through: randomization

- Degenerate feedback loops increase output homogeneity
- Combat homogeneity by introducing *randomness* in predictions
- e.g. recsys: show users random items & use feedback to determine items' quality



Data distribution shifts

Supervised learning: P(X, Y)

- Source distribution: data the model is trained on
- Target distribution: data the model runs inference on

"Testing in Deployment Mode"

1. P(X, Y) = P(Y|X)P(X)
2. P(X, Y) = P(X|Y)P(Y)

Types of data distribution shifts

Type	Meaning	Decomposition
Covariate shift	• P(X) changes • P(Y|X) remains the same	P(X, Y) = P(Y|X)P(X)
Label shift	• P(Y) changes • P(X|Y) remains the same	P(X, Y) = P(X|Y)P(Y)
Concept drift	• P(X) remains the same • P(Y|X) changes	P(X, Y) = P(Y|X)P(X)

Covariate shift

- P(X) changes
- P(Y|X) remains the same

- *Statistics*: a covariate is an independent variable that can influence the outcome of a given statistical trial.
- Supervised ML: input features are covariates Simply the Input Feature
- Input distribution changes, but for *a given input*, output is the same
- E.g. suppose a model is trained to predict whether a patient has a **disease (Y)** based on their **blood test results** (X), such as blood sugar levels and cholesterol levels. The model was trained on **a dataset of patients from a general hospital.** Now, the same model is applied to **a different hospital in a different region, where people have different average diets and lifestyles.** P(x) Changed

Output in the Disease

Covariate shift: (another) example

- P(X) changes
- P(Y|X) remains the same

- Predicts P(cancer | patient) data

Tiny:

- P(age > 40): training > production
- P(cancer | age > 40): training = production

→ A lot of Women in the Training

Covariate shift: causes (training)

- P(X) changes
- P(Y|X) remains the same

- Predicts P(cancer | patient)
- P(age > 40):
  - training > production
- P(cancer | age > 40):
  - training = production

- Data collection
  - E.g. women >40 are encouraged by doctors to get checkups
  - Closely related to sampling biases
- Training techniques
  - E.g. oversampling of rare classes
- Learning process
  - E.g. active learning

Covariate shift: causes (production)

- P(X) changes
- P(Y|X) remains the same

Changes in environments

- Ex 1: P(convert to paid user | free user)
  - New marketing campaign attracting users from with higher income
    - P(high income) increases
    - P(convert to paid user | high level) remains the same
- Ex 2: P(flu | coughing sound)
  - Training data from clinics, production data from phone recordings
    - P(coughing sound) changes
    - P(flu | coughing sound) remains the same

Covariate shift

- Development: if knowing in advance how the production data will differ from training data, use importance weighting
- Production: unlikely to know how a distribution will change in advance

Tricky

Label shift

The Output Distribution Change

- P(Y) changes
- P(X|Y) remains the same

- Output distribution changes but for *a given output*, input distribution stays the same.
- E.g. a disease prediction model trained on the general population but deployed in a hospital where disease is much more common.
  The symptoms **(features; X)** for a disease **(Y)** stay the same, but the overall proportion of sick patients (**P(Y)**) is higher in that hospital.

The Problem

Label shift & covariate shift

- Predicts P(cancer | patient)
- P(age > 40): training > production
- P(cancer | age > 40): training = production
- P(cancer): training > production
- P(age > 40 | cancer): training = prediction

- P(X) changes
- P(Y|X) remains the same

- P(Y) changes
- P(X|Y) remains the same

P(X) change often leads to P(Y) change, so covariate shift often means label shift

Label shift & covariate shift

- Predicts P(cancer | patient)
- New preventive drug: reducing P(cancer | patient) for all patients

- P(age > 40): training > production
- P(cancer | age > 40): training = production
- P(cancer): training > production
- P(age > 40 | cancer): training = prediction

- P(X) changes
- ~~P(Y|X) remains the same~~

↓ So All Woman

- P(Y) changes
- P(X|Y) remains the same

Not all label shifts are covariate shifts!

Concept Drift

Constant

- P(X) remains the same
- P(Y|X) changes

- "Same input, expecting different output"
- P(houses in SF) remains the same
- Covid causes people to leave SF, housing prices drop
  - P($5M | houses in SF)
    - Pre-covid: high
    - During-covid: low
- Concept drifts can be cyclic & seasonal
  - Ride sharing demands high during rush hours, low otherwise
  - Flight ticket prices high during holidays, low otherwise

Nothing Changed in the Input Data (House Record the Same, People Record the Same)

General data changes

- Feature change
  - A feature is added/removed/updated
- Label schema change
  - Original: {"POSITIVE": 0, "NEGATIVE": 1}
  - New: {"POSITIVE": 0, "NEGATIVE": 1, "NEUTRAL": 2}

Detecting data distribution shifts

How to determine that two distributions are different?

1. Compare statistics: mean, median, variance, quantiles, skewness, kurtosis, ...
   - Compute these stats during training and compare these stats in production
   - Not universal: only useful for distributions where these statistics are meaningful
   - Inconclusive: if statistics differ, distributions differ. If statistics are the same, distributions can still differ.

Can be Inconclusive

Cumulative vs. sliding metrics

- Sliding: reset at each new time window

Small Window

Performance of the model

Temporal

This image is based on an example from MadeWithML (Goku Mohandas).

Detecting data distribution shifts

How to determine that two distributions are different?

1. Compare statistics: mean, median, variance, quantiles, skewness, kurtosis, ...
2. Two-sample hypothesis test
   - Determine whether the difference between two populations is statistically significant
   - If yes, likely from two distinct distributions

E.g.
1. Data from yesterday
2. Data from today

Two-sample tests

Drift Detection

Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Kolmogorov-Smirnov	✓	✓		✓	✓		✓
Maximum Mean Discrepancy	✓	✓		✓	✓	✓	
Learned Kernel MMD	✓	✓		✓	✓		
Least-Squares Density Difference	✓	✓		✓	✓	✓	
Chi-Squared	✓				✓		✓
Mixed-type tabular data	✓				✓		✓
Classifier	✓	✓	✓	✓	✓		
Spot-the-diff	✓	✓	✓	✓	✓		✓
Classifier Uncertainty	✓	✓	✓	✓	✓		
Regressor Uncertainty	✓	✓	✓	✓	✓		

alibi-detect (OS)

Most tests work better on low-dim data, so dim reduction is recommended beforehand!

Not all shifts are equal

- Sudden shifts vs. gradual shifts
  - Sudden shifts are easier to detect than gradual shifts
- Spatial shifts vs. temporal shifts

- New device (e.g. mobile vs. desktop)
- New users (e.g. new country)

E.g. same users, same device, but behaviors change over time

Addressing data distribution shifts

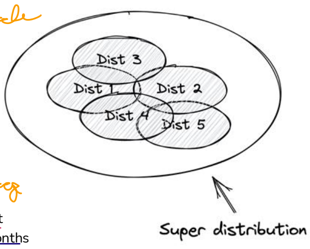
doesn't mean googled

diffent Approach

1. Train model using a massive dataset

   Current Solution

2. Retrain model with new data from new distribution
   - Mode
     - Train from scratch
     - Fine-tune none in language and Image
   - Data
     - Use data from when data started to shift
     - Use data from the last X days/weeks/months
     - Use data form the last fine-tuning point



Need to figure out not just when to retrain models, but also how and what data

Monitoring & Observability

Monitoring vs. observability Closely Related

- Monitoring: tracking, measuring, and logging different metrics that can help us determine when something goes wrong
- Observability: setting up our system in a way that gives us visibility into our system to investigate what went wrong

Monitoring vs. observability

- Monitoring: tracking, measuring, and logging different metrics that can help us determine when something goes wrong
- Observability: setting up our system in a way that gives us visibility into our system to investigate what went wrong

Instrumentation assumes that we have the Set Up the System a Certain Way

- adding timers to your functions
- counting NaNs in your features
- logging unusual events e.g. very long inputs
- ...

Monitoring is all about metrics

- Operational metrics
- ML-specific metrics

Operational metrics How the System Performs

- Latency
- Throughput
- Requests / minute/hour/day
- % requests that return with a 2XX code
- CPU/GPU utilization
- Memory utilization
- **Availability**
- etc.

Service Level Agreement (or KPI)

- Up means:
  - median latency <200ms
  - 99th percentile <2s
- 99.99% uptime (four-nines)

ML metrics: what to monitor

The Input Data

harder to monitor — easier to monitor

raw inputs → features → predictions → accuracy*

Start — Bef Measure

less likely to be caused by human erros

closer to business metrics KPIs

* if natural labels available

Monitoring #1: accuracy-related metrics

exact way

- Most direct way to monitor a model's performance
- Collect as much feedback as possible
- Example: YouTube video recommendations
  - Click through rate
  - Duration watched
  - Completion rate
  - Take rate

Monitoring #2: predictions

- Predictions are low-dim: easy to visualize, compute stats, and do two-sample tests
- Changes in prediction dist. generally mean changes in input dist.
- Monitor odd things in predictions
  - E.g. if predictions are all False in the last 10 mins

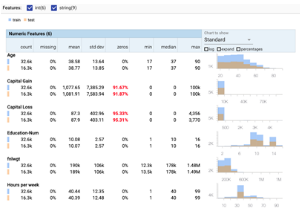
Monitoring #3: features

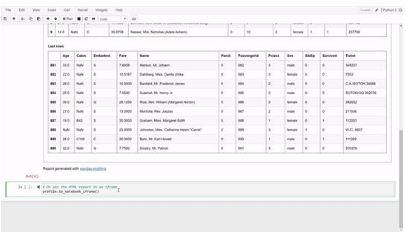
Hard to do

- Most monitoring tools focus on monitoring features
- Feature schema expectations
  - Generated from the source distribution
  - If violated in production, possibly something is wrong
- Example expectations
  - Common sense: e.g. "the" is most common word in English
  - min, max, or median values of a feature are in [a, b]
  - All values of a feature satisfy a regex
  - Categorical data belongs to a predefined set
  - FEATURE_1 > FEATURE_B

Generate expectations with profiling & visualization

- Examining data & collecting:
  - statistics
  - informative summaries
- pandas-profiling
- facets





Monitoring #3: features

- Feature schema expectations

"Unit Testing" for Data Science Applied

Table shape

- `expect_column_to_exist`
- `expect_table_columns_to_match_ordered_list`
- `expect_table_columns_to_match_set`
- `expect_table_row_count_to_be_between`
- `expect_table_row_count_to_equal`
- `expect_table_row_count_to_equal_other_table`

Missing values, unique values, and types

- `expect_column_values_to_be_unique`
- `expect_column_values_to_not_be_null`
- `expect_column_values_to_be_null`
- `expect_column_values_to_be_of_type`
- `expect_column_values_to_be_in_type_list`

```
expect_column_values_to_be_between(
    column="room_temp",
    min_value=60,
    max_value=75,
    mostly=.95
)
```

"Values in this column should be between 60 and 75, at least 95% of the time."

"Warning: more than 5% of values fell outside the specified range of 60 to 75."

GitHub - great-expectations/great_expectations

Feature monitoring problems

1. Compute & memory cost
   a. 100s models, each with 100s features
   b. Computing stats for 10000s of features is costly
2. Alert fatigue
   a. Most expectation violations are benign
3. Schema management
   a. Feature schema changes over time
   b. Need to find a way to map feature to schema version

Monitoring toolbox: logs

- Log everything but only useful rule
- A stream processing problem

we Process

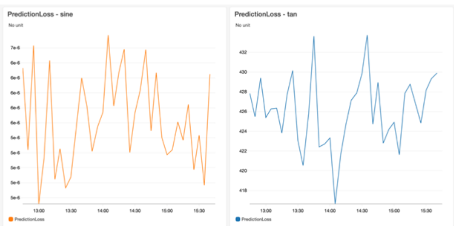
Vladimir Kazanov (Badoo 2019)

"If it moves, we track it. Sometimes we'll draw a graph of something that isn't moving yet, just in case it decides to make a run for it."

Ian Malpass (Etsy 2011)

Monitoring toolbox: dashboards

- Make monitoring accessible to non-engineering stakeholders
- Good for visualizations but insufficient for discovering distribution shifts



Monitoring toolbox: alerts

- 3 components
  - Alert policy: condition for alert
  - Notification channels who do we notify
  - Description
- Alert fatigue
  - How to send only meaningful alerts?

```
## Recommender model accuracy below 90%

${timestamp}: This alert originated from the service ${service-name}
```


Monitoring -> Continual Learning

- Monitoring is passive we Observe if Something goes Wrong
  - Wait for a shift to happen to detect it
- Continual learning is active
  - Update your models to address shifts

Ifect in the Development

Once we detect

We try to Correct