MLOPs

ML Project life cycle

Scoping → Data → Modelling → Deployment

Scoping
- define Project

Data
- define data & establish baseline
- Label & organize data

Modelling
- select & train model
- perform err analysis

Deployment
- Deployment in production
- Monitor & Maintain system

Speech Recognition

Scoping Stage:

- Decide to work on speech recognition for voice search
- Decide on key metrics: • Accuracy, latency, throughput
- Estimate resources & timeline

Data stage:

- Is the data labeled consistently?
- How much silence before/after each clip?
- How to perform volume normalization?

Modelling stage:

Research/Academia { Code (algo/model) → ML Model

Hyperparameters → ML Model

Product Team { Data → ML Model

$\boxed{\text{ML System} = \text{code} + \text{Data}}$

Deployment stage:

challenges in deployment
- concept drift
- data drift

Concept drift & Data drift

- conceptual / methology changes
- changes in Input data

Software Engineering Issues

Checklist of questions

- Realtime or Batch
- Cloud v/s Edge/Browser
- Compute resources (CPU/GPU/memory)
- Latency, througput (QPS)
- Logging
- Security & privacy

Common Deployment Cases

- New product/ capability
- Automate / assist with manual task
- Replace previous ML system

  Key Ideas:
  - Gradual ramp up with monitoring
  - Rollback

Visual Inspection Example

In Mobile factory → Model to find either mobile good or not

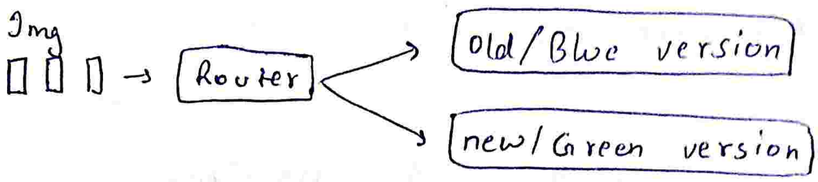
Human	Model
✓	✓
✗	✗
✗	✓

- ML system shadow the human & runs in parallel
- ML system's output not used for any device during this phase.

Canary Deployment

- Roll out to small fraction (say 5%) of traffic initially
- Monitor system & ramp up traffic gradually

Blue - Green Deployment



Easy way to enable rollback.

Degree of Automation

Human Only > Shadow Mode > $\underbrace{\text{AI Assistance} > \text{Partial automation}}_{\text{HITL}}$ > full automation

You can choose to stop before getting to full automation.

Monitoring Dashboard

- It is OK to use multiple metrics initially & gradually remove the ones you find not useful.

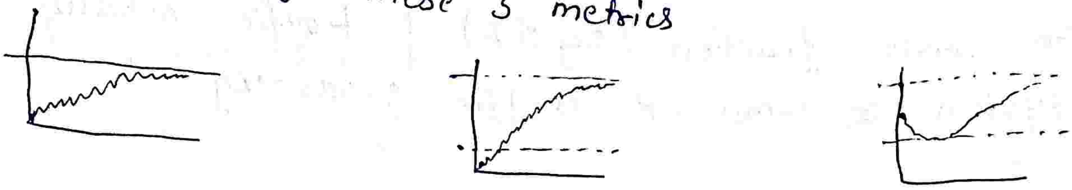
What is Monitoring: It means continuously observing & checking how your machine learning system is working after deployment.

What do we monitor in ML systems?

- Model Performance metrics :① Accuracy, etc. ① compare with baseline performan.
- Data Quality : Are values in expected range?
- System health : Latency, Availability, Resources
- Data/Concept Drift.

For Monitoring Methods

- Let we using these 3 metrics



- So we do these 2 things
  1) Set thresholds for alarms
  2) Adapt metrics & thresholds over time

Model Maintenence

- manual retraining
- Automatic retraining

Metrics to monitor

Monitor : software metrics, Input metrics & output metrics

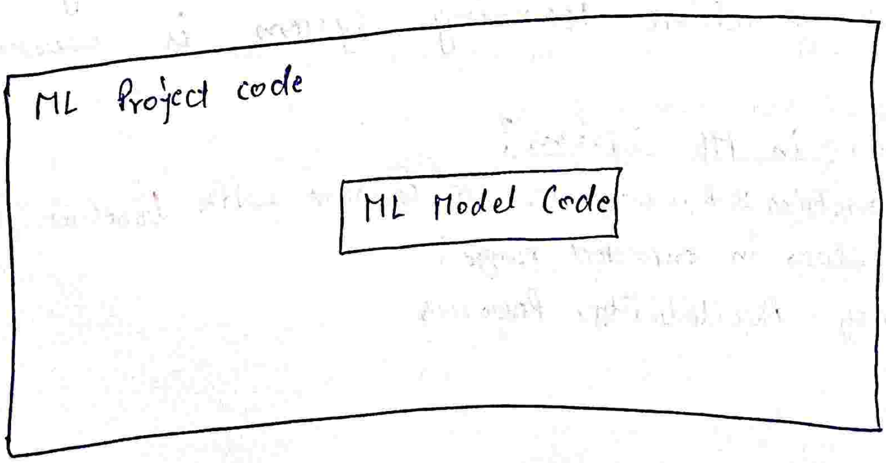
How Quickly do they changes ?

- user data generally has slower drift.
- Enterprise data (B2B app) can shift fast.

Challenges in model development

- Doing well on training set (usually measured by avg training e
- Doing well on dev/test set
- Doing well on business metrics/project goals.

ML in Production



#11

Select & train Model

Why low avg test error isn't good enough

- Performance on disproportionately : ex: web search engine
- Performance on key slices of the dataset
  ex. - ml for loan approval
  - Product recommendations from retailers
- Rare classes
  - skewed data distribution {medical diagnose example}

Establish a baseline

Speech recognition example

Type	Accuracy	Human level Performance		
Clear speech	94%	95%	⟶	1%
Car Noise	89%	93%	⟶	4%
People Noise	87%	89%	⟶	2%
Low Bandwidth	70%	70%	⟶	~0%

Ways to establish baseline

- Human Level Performance (HLP)
- Literature search for state-of-the-art / open source
- Quick-and-dirty implementation
- Performance of older system

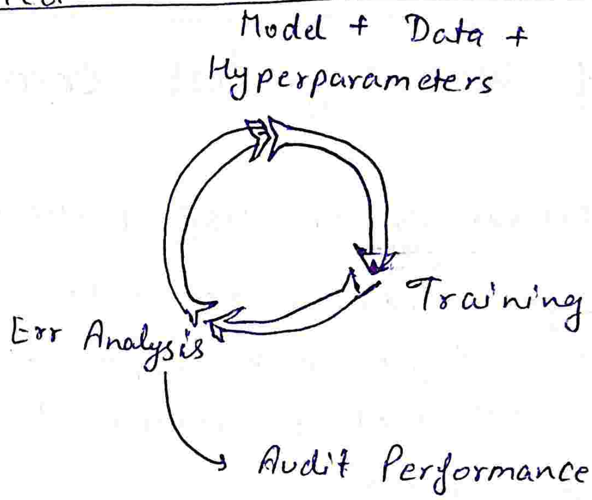
Baseline helps to indicates what might be possible. In some cases (such as HLP) is also gives a sense of what is irreducible error/Bayes error.

↓

कभी भी Reduce ना होने वाला err

Tips for getting started

• ML is iterative process



Getting started on Modeling

• Literature search to see what's possible (courses, blog, open-src projects)
• Find oper-src implementations if available.
• A reasonable algorithm with good data will often outperform a great algorithm with ~~so~~ so good data

Deployment Constraints when picking a model

Q- Should you take into account deployment constraint when picking a model?

As → Yes, if baseline is already established and goal is to build and deploy.
No(or not necessarily), if purpose is to establish a baseline and determine what is possible and might be worth pursuing

Quick & simple test
↓
☆ Sanity-check for code & algorithm
- Try to overfit a small training dataset before training on a large one.

Error analysis & Performance Auditing

~~Example~~ How to do err analysis

1) collect wrong predictions 2) Tag the errors 3) count & analyze

→ Err.A is Iterative process
→ helps to focus on right improvement

Prioritizing what to work on

Don't just fix the biggest errors, instead, prioritize based on impact (size x frequency) and importance to business goals.

Type	Accuracy	Human level Performance	~~Gap~~ Gap to HLP	% of data	
clean speech	94%	95%	1%	60%	→ 0.6%
car Noice	89%	93%	4%	4%	→ 0.16%
People Noice	87%	89%	2%	30%	→ 0.6%
Low Bandwidth	70%	70%	0%	6%	→ ~0%

Skewed Datasets

- for skwed d. use confusion metric for error analysis or its metrics like Precision, Recall, F1-score

Perfor. Auditing

Even model shows good accuracy/F1-score, bef before deploy it to production we must double check it (audit) to make sure it works. fairly, reliably, and safely.

It means the model on different slices of data, checking for bias, fairness, rare cases, and aligning with business expectations.

▷ Evaluate & audit with the business team.

Data Iteration

Data Centric AI Development

- newer approach, focusing on improving data quality, not just model
- Error Analysis
- Data Augmentation
- Label Cleaning
- Balanced sampling

Model Centric AI Development

- traditional way
- here we fix dataset & focus on improving the model
- Data is constant (eg. benchmark dataset, like MNIST or CIFAR-10)

Data Iteration loop : Instead of doing model iteration (train → adjust model → retrain), we do a data iteration loop.

Data → Model → Err Analysis → Improve Data → Repeat

Data Augmentation

Goal : Create ~~real~~-realistic examples that
(i) the algo does poorly on, but
(ii) humans (or other baseline) do well on

Check List : According to speech recog. example

- ☐ Does it sound realistic?
- ☐ Is the $x \to y$ mapping clear? (eg. can humans recognize speech?)
- ☐ Is the algo currently doing poorly on it?

Tips :
- Don't overdo → unrealistic data can hurt
- Useful to target weak spots found in err analysis
- Large models tolerate distribution shifts better than small models

Can adding data hurt?

- usually adding data helps, but rare cases can hurt.
- when it's safe :
  - Large models (high capacity)
  - clear X → Y mapping (labels not ambiguous)
- Risky when
  - small model → over focus on oversampled class
  - Labels ambiguous (eg. digit "1" vs letter "l"). Too much augmentation of ambiguous cases can confuse model.

Adding Features (Structured Data)

- For structured data, generating new training examples is hard.
- Instead → add new features
- Example : Restaurant Recommendation system :-

  Issue : vegetarians recommended meat-only restaurants

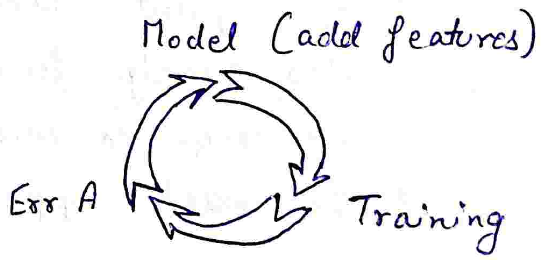
  Fix : Add features like "% vegetarian meals ordered" (user) + "restaurants has veg options" (restaurant)

- Features can be hand-coded or learned automatically.
- Trend : shift from collaborative filtering (similar users) → content based filtering (use item/user features)
- Helps with cold start Problem (new product/restaurant)

Data Iteration for Structured Data

- Err A. can be harder if there is no good baseline (like HLP) to compare to

  or competitor benchmarking



Experiment Tracking

- Crucial when running many experiments.
- Track :
  - Algorithm/core versions
  - Dataset used
  - Hyperparameters
  - Results (metrics + ideally save trained model)
- Tools :
  - Basis : Text files → spreadsheets
  - Advanced : Weights & Bias, Comet, MLflow, SageMaker Studio
- Good tracking helps with:
  - Replicability (same code/data → same result)
  - Efficiency (don't repeat failed experiments)
  - Analysis (see which settings worked)
- Internet fetched data changes → hurts replicability.

From Big Data to Good Data

- Try to ensure consistently high-quality data in all phase of ML project lifecycle

Good Data

- Covers important cases (good coverage of inputs x)
- Is defined consistently (definition of labels y is unambiguous)
- Has timely feedback from production data (distribution covers data drift and concept drift)
- Is sized appropriately

#26

Define Data and establish Baseline

☆ Data Definition Questions

- What is the Input x?
  - Lighting? Contrast? Resolution?
  - What features need to be included?
- What is the target label y?

☆ Major types of Data Problems

	Unstructured	Structured	
Small Data	Manufacturing visual inspection from 100 training examples	Housing price prediction based on square footage, etc. from 50 training examples	≤ 10000 Clean labels are critical
Big Data	Speech recognition from 50 million training examples	Online shopping recommendations for 1 million users	> 10000 Emphasis on data process

Unstructured:
- Humans can label data
- Data Aug

Structured: Harder to obtain more data

☆ Unstructured V/s Structured data

Unstructured data

- May or may not have huge collection of unlabeled examples x.
- Human can label more data
- Data Aug. more likely be helpful

Structured data

- May be more difficult to obtain more data
- Human labeling may not be possible (with some exceptions)

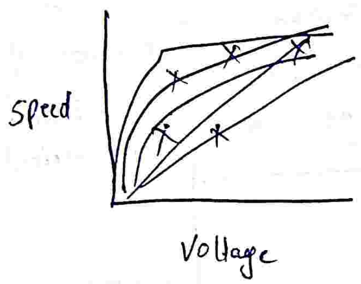
☆ Small Data v/s Big Data

Small Data
- clean labels are critical
- Can manually look through dataset & fin labels
- can get all the labelers to talk to each other

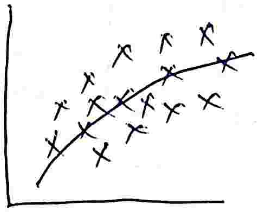
Big Data
- Emphasis data process.

☆ Small Data and label consistency

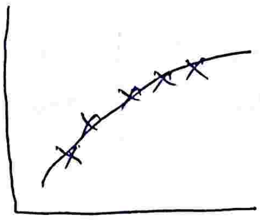
Why label consistency is important



- Small data
- Noisy labels



- Big data
- Noisy labels



- small data
- Clean (consistent) labels

Big data problems can have small data challenges too

- Problem with large dataset but where there's a long tail of rare events in the input will have small data challenges too.
  - Web search
  - Self-driving cars
  - Product Recommendation systems

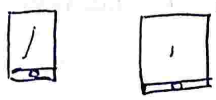
☆ Improving labels consistency

- Have multiple labelers label same example
- When there is disagreement, have MLE, subject matter expert (SME) and/or labelers discuss definition of y to reach agreement.
- If labelers believe that x doesn't contain information, consider changing x.
- Iterate until it is hard to significantly increase agreement.
- Examples
  - $\boxed{\text{Standardize labels}}$

    "Um, nearest gas station"
    "Umm, nearest gas station"
    "Nearest gas station [unintelligible]"

    ⇒ "Um, nearest gas station"

  - $\boxed{\text{Merge classes}}$



Deep scratch   Shallow scratch

⇒ Scratch

☆ small data v/s big data (unstructured)

Small Data

- usually small number of labelers
- can ask labelers to discuss specific labels

Big Data

- Get to consistent definition with a small group
- Then send labelling instructions to labelers
- Can consistent having multiple labelers label every example and using voting or consensus labels to increase accuracy.

☆ Human Level Performance

Ground ~~Level~~ Truth Label	Inspector
1	1
1	0
1	1
0	0
0	0
0	1

→ 66.7% accuracy

Why measure HLP?

Estimate Bayes error/irreducible err to help with err A. and prioritization

Other uses of HLP

- In academia, establish and beat a respectable benchmark to support publication.
- Business or product owner asks for 99% accuracy. HLP helps establish a more reasonable target.
- "Prove" the ML system is superior to humans doing the job and thus the business or product owner should adopt it

↑ use with caution

☆ Rising HLP

- When the ground truth label is externally defined, HLP gives an estimate for Bayes err/irreducible err
- But often ground truth is just another human label.

Scratch length (mm)	Ground Truth Label	Inspector
0.7	1	1
0.2	~~1~~ 0	0
0.5	1	1
0.2	0	0
0.1	0	0
0.1	0	~~1~~ 0

66.7% ↓ 100%

- when the label y comes from a human label, HLP << 100% may indicate ambiguous labeling instructions Um... Umm...
- Improving label consistency will raise HLP
- This makes it harder for ML to beat HLP. But the more consistent labels will raise ML performance, which is ultimately likely to benefit the actual application performance.

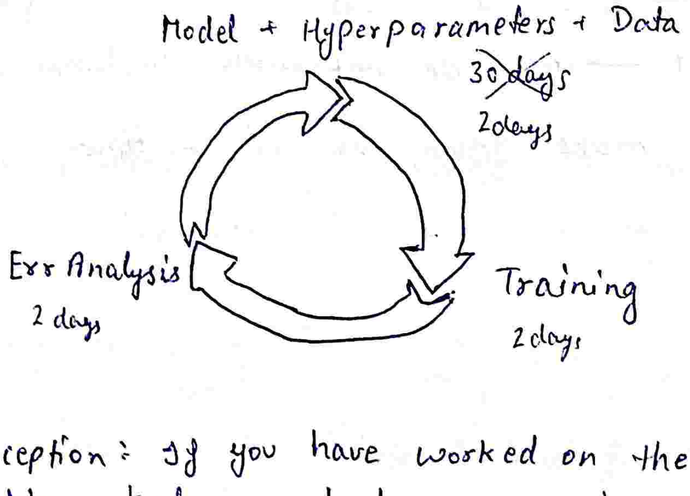
HLP on structured data

Structured data problems are less likely to involve human labelers, thus HLP is frequently used.

Some exceptions:
- User ID merging: Same person?
- Based on network traffic, is the computer hacked?
- Is the transaction fraudulent?
- Spam account? Bot?
- ~~Gf~~ From GPS, what is the mode of transportation — on foot, bike, car, bus?

#32

Obtaining Data



- Exception: If you have worked on the problem before and from experience you know you need m examples.

How long should you spend obtaining data?

- Get into this iteration loop as quickly possible.
- Instead of asking: How long it would take to obtain m examples?
  Ask: How much data can we obtain in k days?
- ~~Exception: If you have~~

Inventory Data

Brainstorm list of data sources (speech recognition)

Source	Amount	Cost	Time
owned	100 h	₹0	0
Crowdsourced - Reading	1000 h	₹10 000	14 days
Pay for labels	100h	₹ 6 000	7 days
Purchase data	1000 h	₹ 10 000	1 day

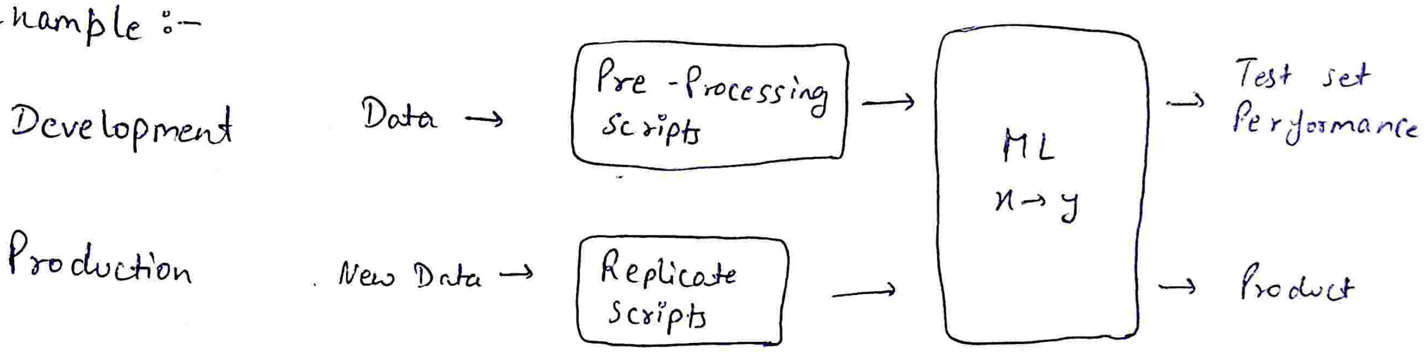
other factors : Data quality, privacy, regulators constraints

Labeling Data

- options : In-house v/s outsourced v/s crowdsourced
- Having MLEs label Data is expensive. But doing this for just a few days is usually fine
- who is qualified to label ?
  - speech recognition — any reasonably fluent speaker
  - Factory inspection, medical image diagnosis — SME (Subject Matter Expert)
  - Recommender systems — may be impossible to label well
- Don't increase data by more than 10x at a time.

Data Pipeline

Example :-



POC and Production Phase

POC (Proof-of-concept) :

- Goal is to decide if the application is workable & worth deploying.
- Focus on getting the prototype to work!
- It's ok if data pre-processing is manual. But take extensive notes/comments

Production Phase:

- After project utility is established, use more sophisticated tools to make sure the data pipeline is replicate
- E.g., TensorFlow Transform, Apache Beam, Airflow, ...

Meta Data, Data-Provenance and Lineage

Task: Predict if someone is looking for a Job

x = user data, y = looking for a job

Keep track of data provenance and lineage

provenance → where it comes from

lineage → Sequence of steps