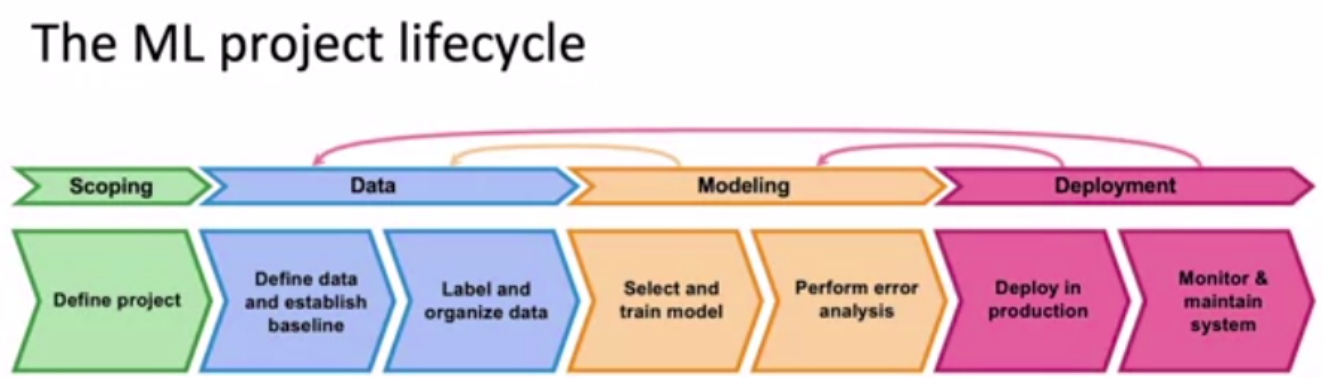
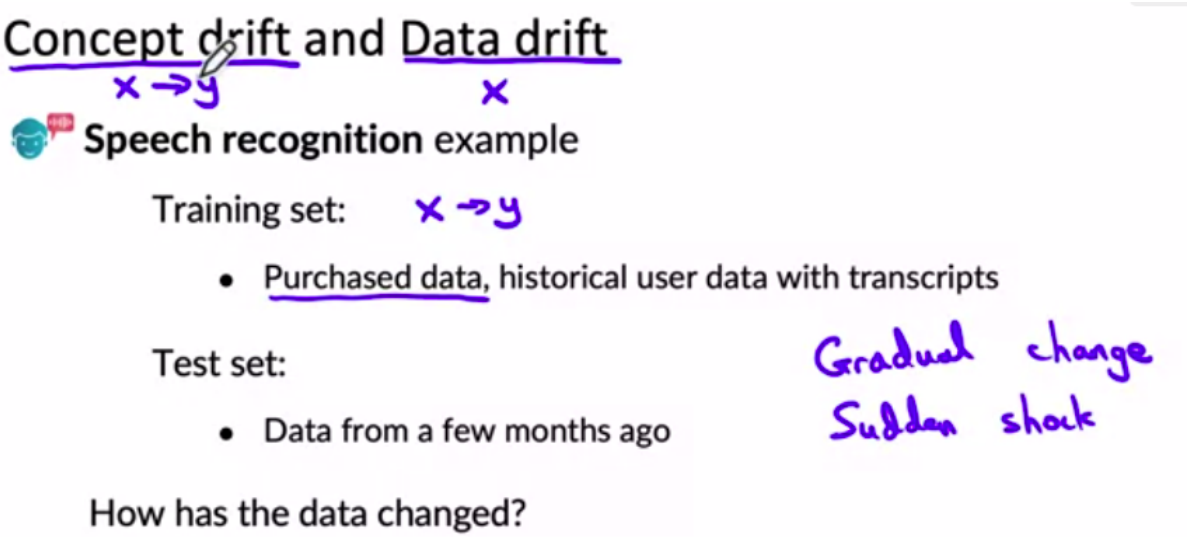
Introduction to Machine Learning in Production

Week 1:



Deployment:



When data changes in general, this problem is called **data drift** whereas the changes in the context of the target variable are called **concept drift**.Both of these drift **causes model decay** but needs to be addressed separately.

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.

Training-serving skew

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.

Concept drift

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

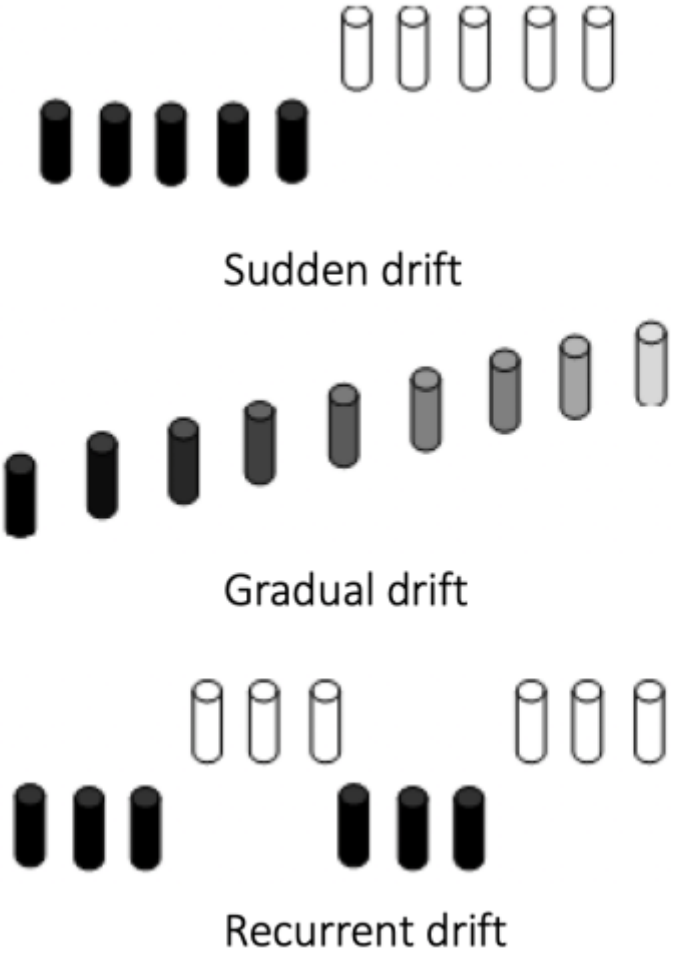
In contrast to the data drift, the distributions (such as user demographics, frequency of words, etc.) might even remain the same. Instead, the relationships between the model inputs and outputs change.

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.

Model decay could occur due to the following type of shift.

- Covariate Shift: Shift in the independent variables.
- Prior Probability Shift: Shift in the target variable.
- Concept Drift: Shift in the relationship between the independent and the target variable.

Primarily there are 3 kinds of concept drift as depicted in Fig and each type of phenomenon requires a different method to detect it (and monitor the change).



- Sudden: Where the concept drift happens abruptly due to unforeseen circumstances like COVID-19 pandemic which affected several sectors like eCommerce, health care, finance, insurance, and many more. Such abrupt change might happen in as short as a few weeks. This kind of drift is usually driven by some external event. If there is no active monitoring system to detect drift in data, it's natural to perform a quick evaluation of the presence of concept drift after a major event.
- Gradual: This kind of drift takes a long time to occur and for many use-cases, it's quite natural to happen. As an example, inflation can affect a pricing model which might take a long time to put a significant impact. Gradual or sometimes called incremental changes, are usually addressed in time series model by capturing the change in seasonality, if not addressed, this is a matter of concern and needs to be addressed.

- Recurrent: This kind of drift happens periodically, maybe during a specific time in a year. As an example, during events like Black Friday, Halloween, etc, users' shopping pattern is different compared to other times in the year. So, a different model specifically trained on the Black Friday data is used during the time of that event. Recurrent patterns are difficult to monitor as the periodicity of a pattern might also be dynamic.

Based on the type of concept drift that exists, there is a diverse set of methods to monitor concept drift.

Some of the most prominent methods are:

- Monitoring the performance of the model over a long time. As an example, one might monitor F1-score as an accuracy metric and if the score is deteriorating over a long time then this could be a signal of concept drift.
- Monitoring the classification confidence (applicable only to classification). The confidence score of a prediction reflects the probability of a data point belonging to the predicted class. *A significant difference in the average confidence score in two windows represents the occurrence of concept drift.*

Dealing with drift

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.

Software engineering issues

Checklist of questions

- Realtime or Batch
- Cloud vs. Edge/Browser
- Compute resources (CPU/GPU/memory)
- Latency, throughput (QPS)
- Logging
- Security and privacy

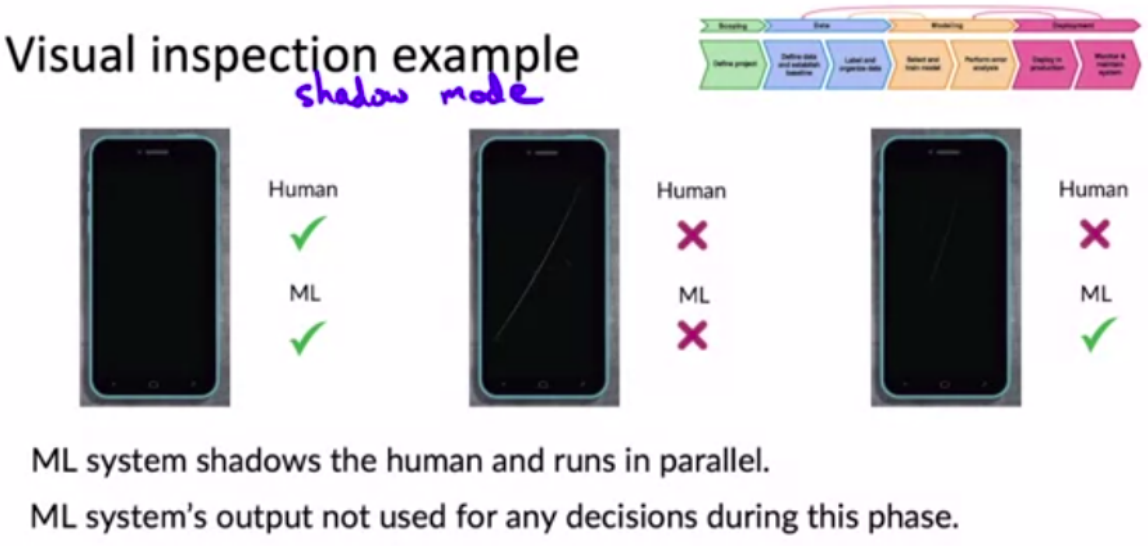
500ms, 1000 QPS

Common deployment cases

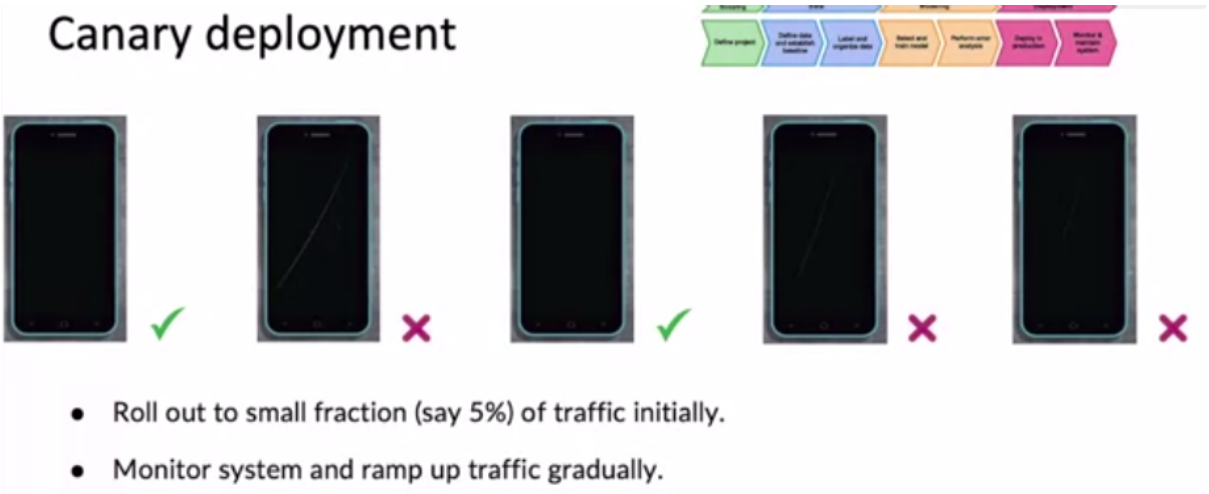
1. New product/capability
2. Automate/assist with manual task
3. Replace previous ML system

Key ideas:

- Gradual ramp up with monitoring
- Rollback

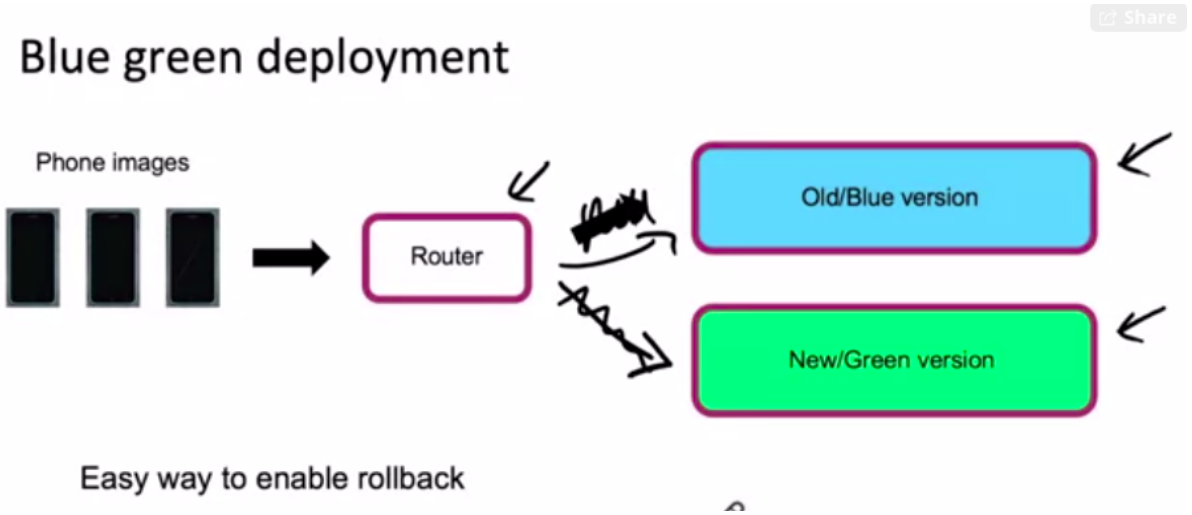


"Shadow Mode" or "Dark Launch" as Google calls it is a technique where production traffic and data is run through a newly deployed version of a service or machine learning model, without that service or model actually returning the response or prediction to customers/other systems. Instead, the old version of the service or model continues to serve responses or predictions, and the new version's results are merely captured and stored for analysis.



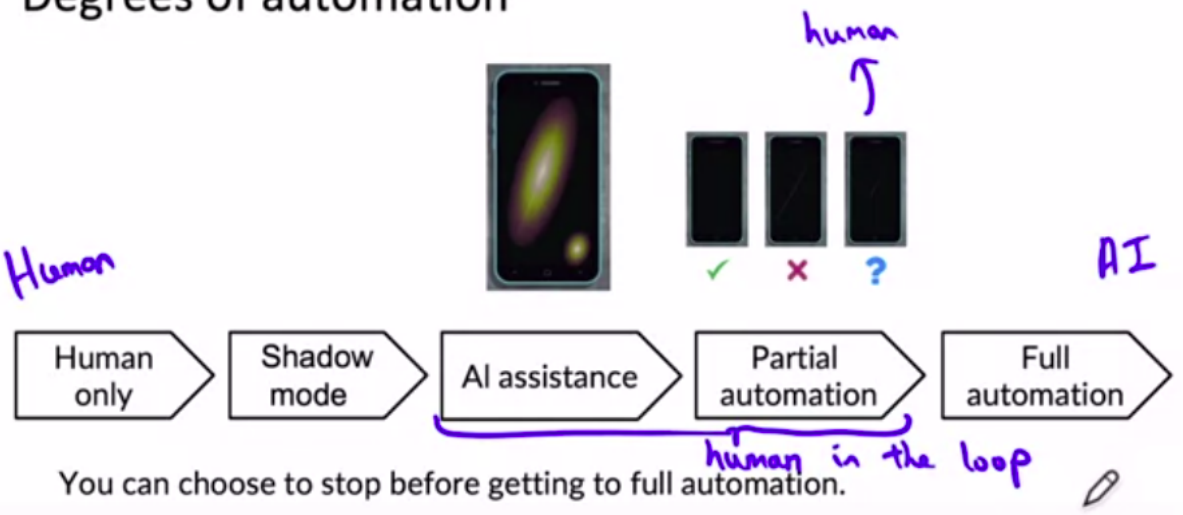
Canary deployments are a pattern for rolling out releases to a subset of users or servers. The idea is to first deploy the change to a small subset of servers, test it,

and then roll the change out to the rest of the servers. The canary deployment serves as an early warning indicator with less impact on downtime: if the canary deployment fails, the rest of the servers aren't impacted.



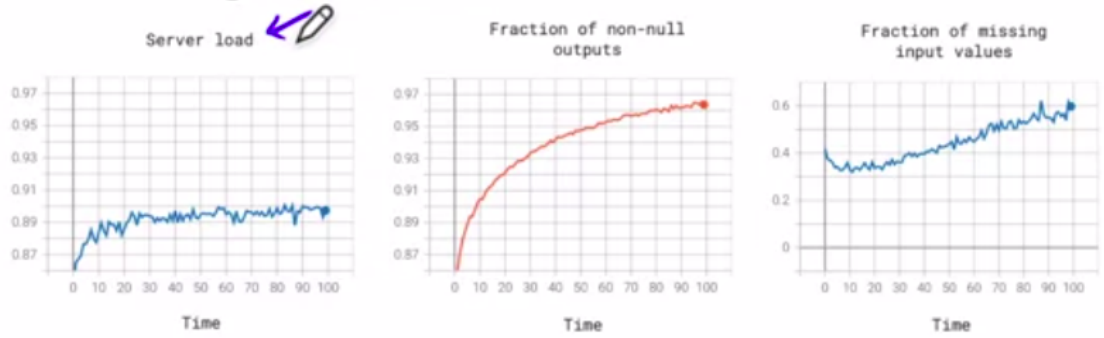
Blue green deployment is an application release model that gradually transfers user traffic from a previous version of an app or microservice to a nearly identical new release—both of which are running in production. The old version can be called the blue environment while the new version can be known as the green environment. Once production traffic is fully transferred from blue to green, blue can standy in case of rollback or pulled from production and updated to become the template upon which the next update is made.

Degrees of automation



You can choose to stop before getting to full automation.

Monitoring dashboard



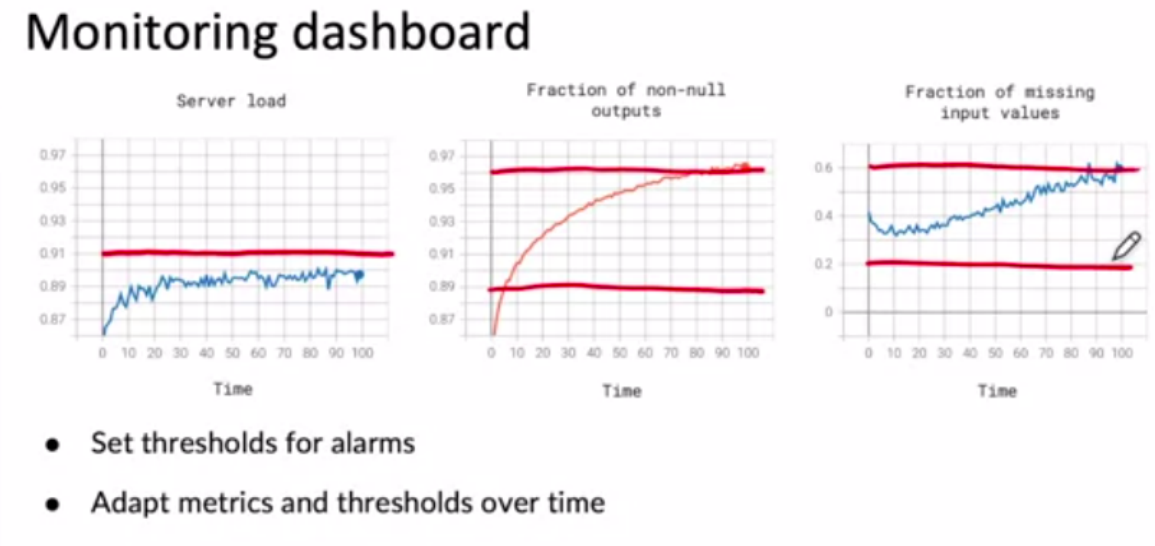
- Brainstorm the things that could go wrong.
- Brainstorm a few statistics/metrics that will detect the problem.

Examples of metrics to track

Software metrics:	Memory, compute, latency, throughput, server load
Input metrics: x	Avg input length Avg input volume Num missing values Avg image brightness
Output metrics: y	# times return " " (null) # times user redoes search # times user switches to typing CTR

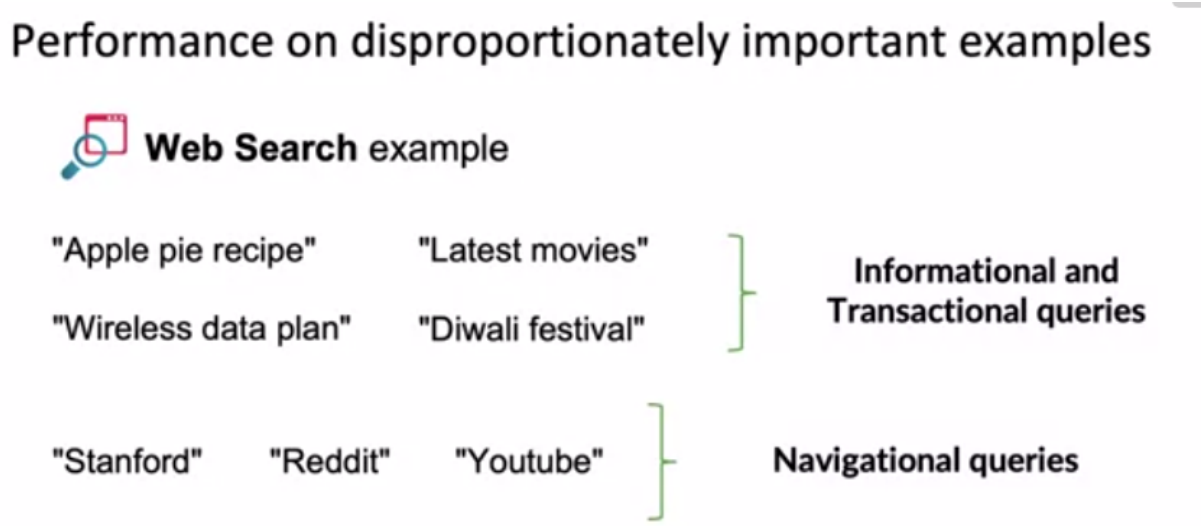
Just as ML modeling is iterative, so is deployment

Iterative process to choose the right set of metrics to monitor.



Week 2:

Selecting and Training Model



It is important to give accurate answers to navigational queries since we risk losing a lot of users if that step isn't carried out efficiently. People just want to navigate to a specific website, so it's important that our top result come out as a website URL and nothing else. This is not the case for informational and transactional queries.

Performance on key slices of the dataset

Example: ML for loan approval

Make sure not to discriminate by ethnicity, gender, location, language or other protected attributes.

Example: Product recommendations from retailers

Be careful to treat fairly all major user, retailer, and product categories.

Establishing a baseline level of performance

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%

HLP

Unstructured and structured data

Unstructured data

Image

Audio

Text

This restaurant was great!

HLP

Structured data

User ID	Purchase	Number	Price
3421	Blue shirt	5	$20
612	Brown shoes	1	$35

Product ID	Product name	Inventory
385	Football	158
477	Cricket bat	23

For unstructured data, we can use HLP as a baseline since humans can very easily give accurate predictions for unstructured data. But for structured data, there need to be some other kind of baseline since humans are not fit for it.

Ways to establish a 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.

Prioritizing what to work on

Decide on most important categories to work on based on:

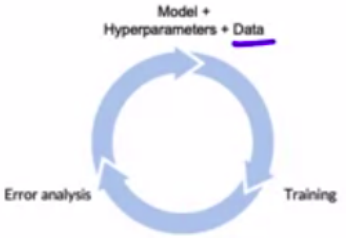
- How much room for improvement there is.
- How frequently that category appears.
- How easy is to improve accuracy in that category.
- How important it is to improve in that category.

Adding/improving data for specific categories

For categories you want to prioritize:

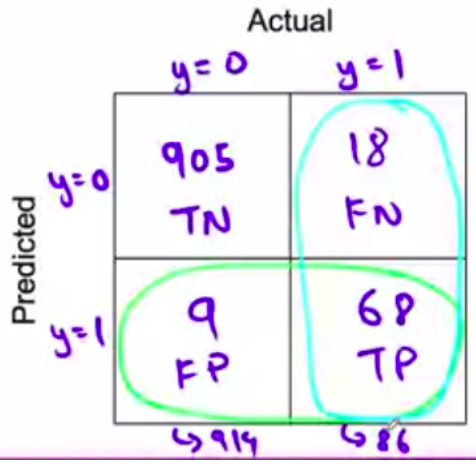
- Collect more data
- Use data augmentation to get more data
- Improve label accuracy/data quality

Type	Accuracy	Human level performance	Gap to HLP	% of data
Clean Speech	94%	95%	1%	60%
Car Noise	84%	93%	4%	4%
People Noise	87%	89%	2%	30%
Low Bandwidth	70%	70%	0%	6%



% of data => how much % of data of that specific category is in the whole dataset. There is not much improvement we can do in case of Car Noise, since total improvement = 4% x 4% = 0.16% which is much lesser than Clean speech (0.6%) and People noise (0.16%)

Confusion matrix: Precision and Recall



TN: True Negative

TP: True Positive

FN: False Negative

FP: False Positive

$\text{Precision} = \frac{TP}{TP+FP} = \frac{68}{68+9} = 88.3\%$

$\text{Recall} = \frac{TP}{TP+FN} = \frac{68}{68+18} = 79.1\%$

Combining precision and recall – F_1 score

	Precision (P)	Recall (R)	F_1
Model 1	88.3	79.1	83.4% ←
Model 2	97.0	7.3	13.6%

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Auditing framework

Check for accuracy, fairness/bias, and other problems.

1. Brainstorm the ways the system might go wrong.
   - Performance on subsets of data (e.g., ethnicity, gender).
   - How common are certain errors (e.g., FP, FN).
   - Performance on rare classes.
2. Establish metrics to assess performance against these issues on appropriate slices of data.

Speech recognition example

1. Brainstorm the ways the system might go wrong.
   - Accuracy on different genders and ethnicities.
   - Accuracy on different devices.
   - Prevalence of rude mis-transcriptions.

GAN gan gang

2. Establish metrics to assess performance against these issues on appropriate slices of data.
   - Mean accuracy for different genders and major accents.
   - Mean accuracy on different devices.
   - Check for prevalence of offensive words in the output.

Data-centric AI development

Model-centric view	Data-centric view
Take the data you have, and develop a model that does as well as possible on it.	The quality of the data is paramount. Use tools to improve the data quality; this will allow multiple models to do well.
Hold the data fixed and iteratively improve the code/model.	*Hold the code fixed and iteratively improve the data.*

Data augmentation

Goal:

Create realistic examples that (i) the algorithm does poorly on, but (ii) humans (or other baseline) do well on

Checklist:

- Does it sound realistic?
- Is the x → y mapping clear? (e.g., can humans recognize speech?)
- Is the algorithm currently doing poorly on it?

Can adding data hurt performance?

For unstructured data problems, if:

- The model is large (low bias).
- The mapping x → y is clear (e.g., given only the input x, humans can make accurate predictions).

Then, **adding data rarely hurts accuracy.**

Experiment tracking

What to track?

- Algorithm/code versioning
- Dataset used
- Hyperparameters
- Results

Tracking tools

- Text files
- Spreadsheet
- Experiment tracking system

Desirable features

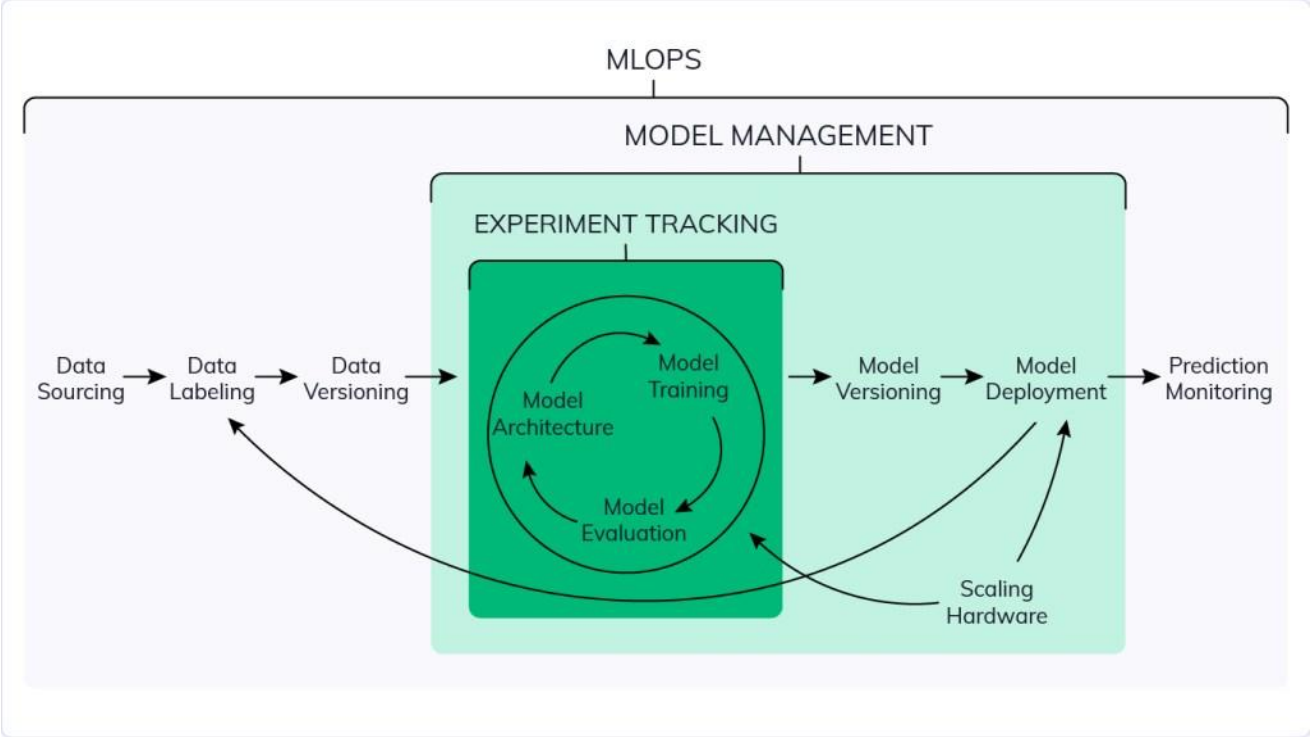
- Information needed to replicate results
- Experiment results, ideally with summary metrics/analysis
- Perhaps also: Resource monitoring, visualization, model error analysis

From Big Data to Good Data

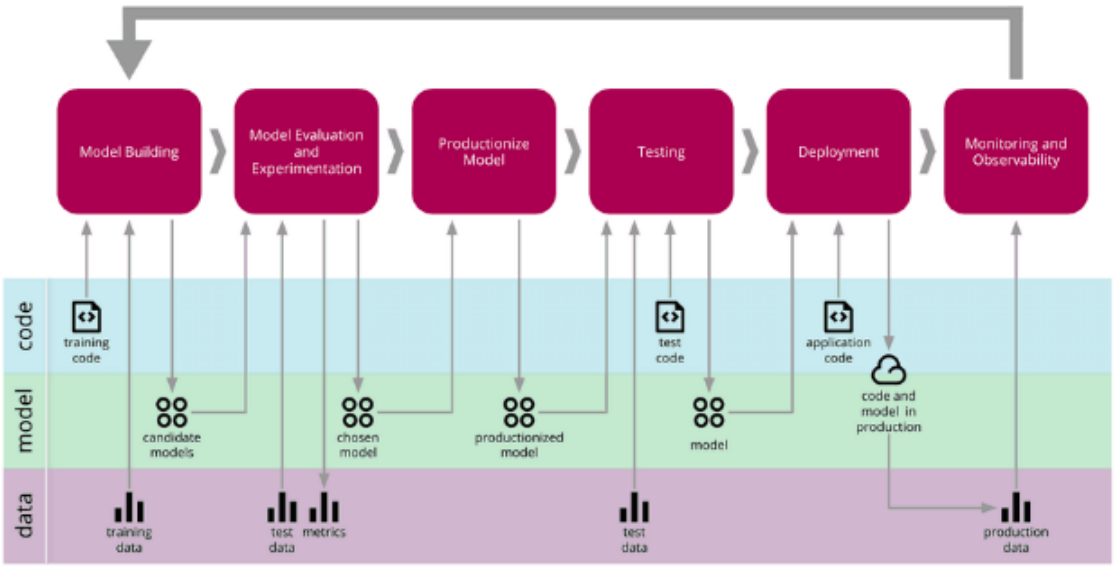
Try to ensure consistently high-quality data in all phases of the ML project lifecycle.

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



Martin Fowler has popularized the concept of Continuous Delivery for Machine Learning (CD4ML), and the diagram for this concept offers a useful visual guide to the ML lifecycle and where monitoring comes into play:

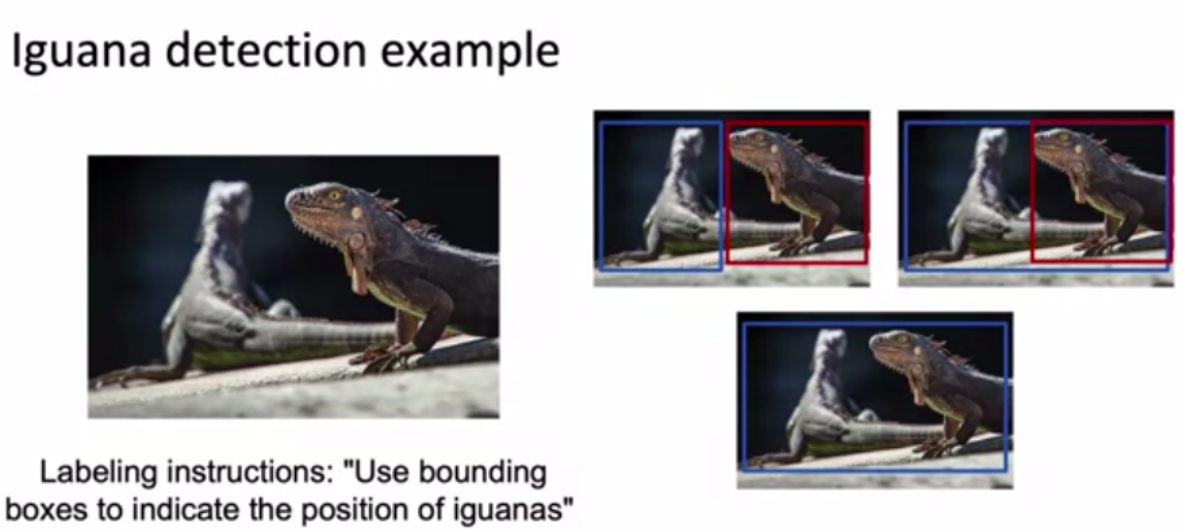


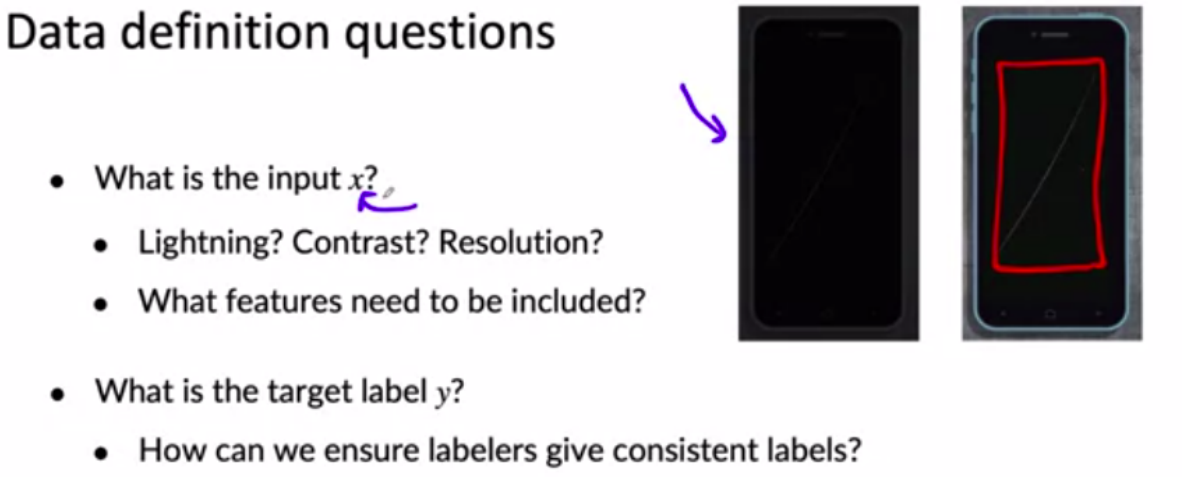
https://christophergs.com/machine%20learning/2020/03/14/how-to-monitor-machine-learning-models/

Week 3:

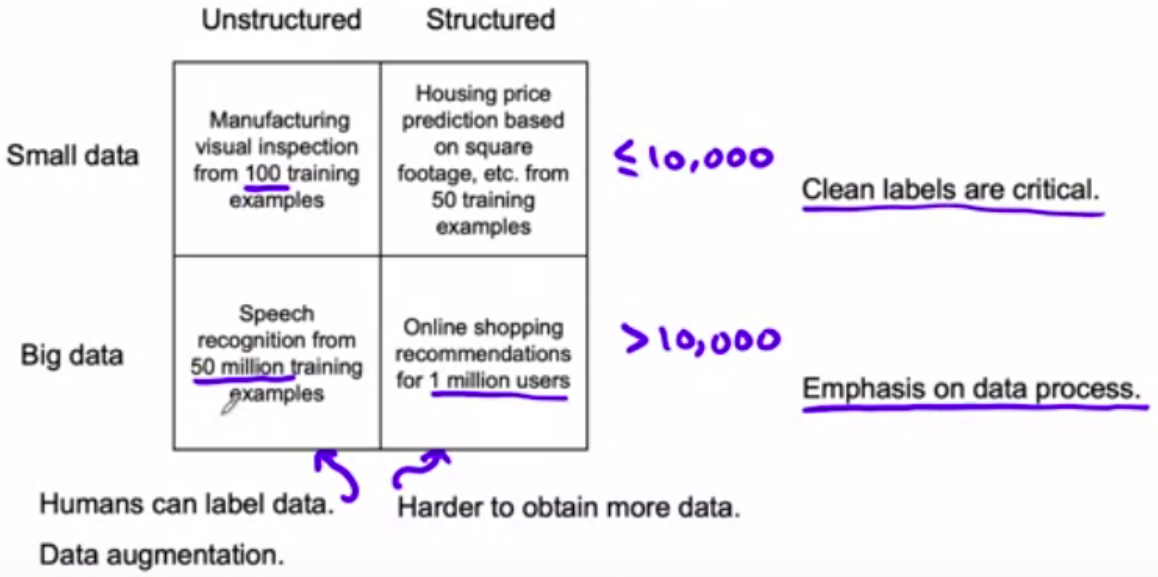
Define data and establish baseline:

Example of inconsistent labelling





Major types of data problems



Unstructured vs. structured data

Unstructured data

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

Structured data

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

Small data vs. big data

< 10,000 > 10,000

Small data

- Clean labels are critical.
- Can manually look through dataset and fix labels.
- Can get all the labelers to talk to each other.

Big data

- Emphasis data process.

Big data problems can have small data challenges too

Problems with a 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 label 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 enough information, consider changing x.
- Iterate until it is hard to significantly increase agreement.

Small data vs. big data (unstructured data)

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 labeling instructions to labelers.
- Can consider having multiple labelers label every example and using voting or consensus labels to increase accuracy.

Why measure HLP?

→ Estimate Bayes error / irreducible error to help with error analysis and prioritization.

99%

Ground Truth Label	Inspector
1	1 ✓
1	0 ✗
1	1 ✓
0	0 ✓
0	0 ✓
0	1 ✗

Human?

66.7% accuracy

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 ✗

Raising HLP

- When the label y comes from a human label, HLP << 100% may indicate ambiguous labeling instructions. Um, Um...
- 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.

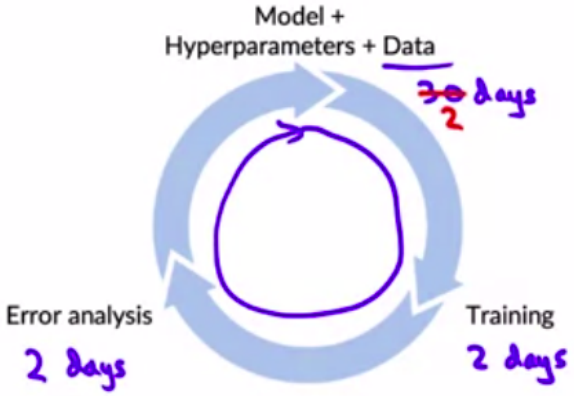
HLP on structured data

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

Some exceptions:

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

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 worked on the problem before and from experience you know you need m examples.

Labeling data

- Options: In-house vs. outsourced vs. 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 – maybe impossible to label well
- Don't increase data by more than 10x at a time

POC and Production phases

POC (proof-of-concept):

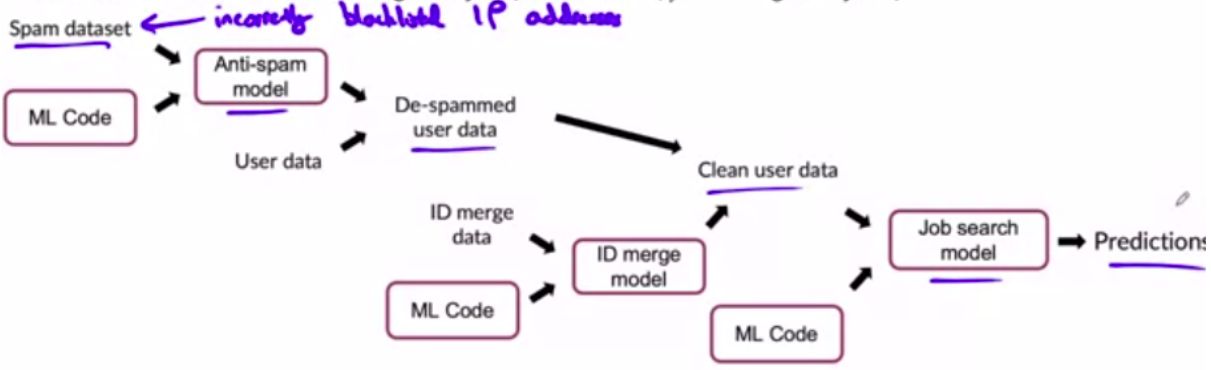
- Goal is to decide if the application is workable and worth deploying.
- Focus on getting the prototype to work!
- It's ok if data pre-processing is manual. But take extensive notes/comments.

Production phase:

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

Data pipeline example

Task: Predict if someone is looking for a job. (x = user data, y = looking for a job?)



Keep track of data provenance and lineage

where it comes from — sequence of steps

After building the pipeline, we realise that there were some incorrectly blacklisted IP addresses, so that would result in us going through the whole pipeline and making changes

again, which is something we shouldn't do and avoid. It's important to store meta-data so that we can go back and see where the error came from and how we can rectify it.

Meta-data

Examples:

- Manufacturing visual inspection: Time, factory, line #, camera settings, phone model, inspector ID,....
- Speech recognition: Device type, labeler ID, VAD model ID,....

Useful for:

- Error analysis. Spotting unexpected effects.
- Keeping track of data provenance.

Balanced train/dev/test splits in small data problems

Visual inspection example: 100 examples, 30 positive (defective)

Train/dev/test: 60% / 20% / 20%

Random split: 21 / 2 / 7 positive example

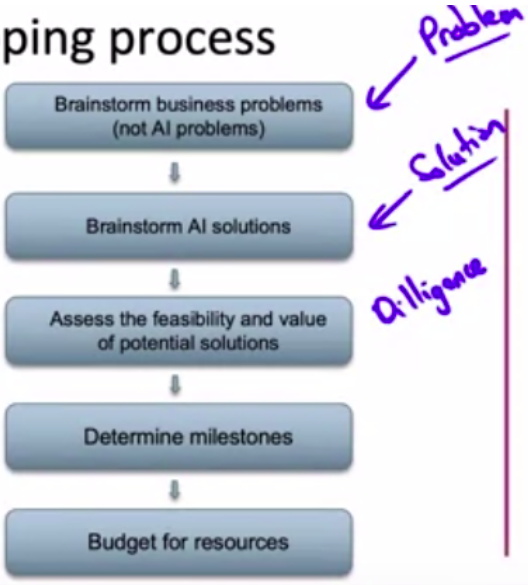
35% 10% 35%

Want: 18 / 6 / 6 } balanced split

30% / 30% / 30%

No need to worry about this with large datasets – a random split will be representative.

Scoping process



What are the top 3 things you wish were working better?

- Increase conversion
- Reduce inventory
- Increase margin (profit per item)

Machine Learning Data Lifecycle in Production

Week 1:

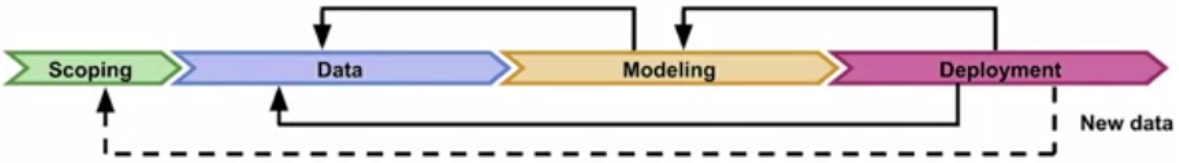
ML modeling vs production ML

	Academic/Research ML	Production ML
Data	Static	Dynamic - Shifting
Priority for design	Highest overall accuracy	Fast inference, good interpretability
Model training	Optimal tuning and training	Continuously assess and retrain
Fairness	Very important	Crucial
Challenge	High accuracy algorithm	Entire system

Challenges in production grade ML

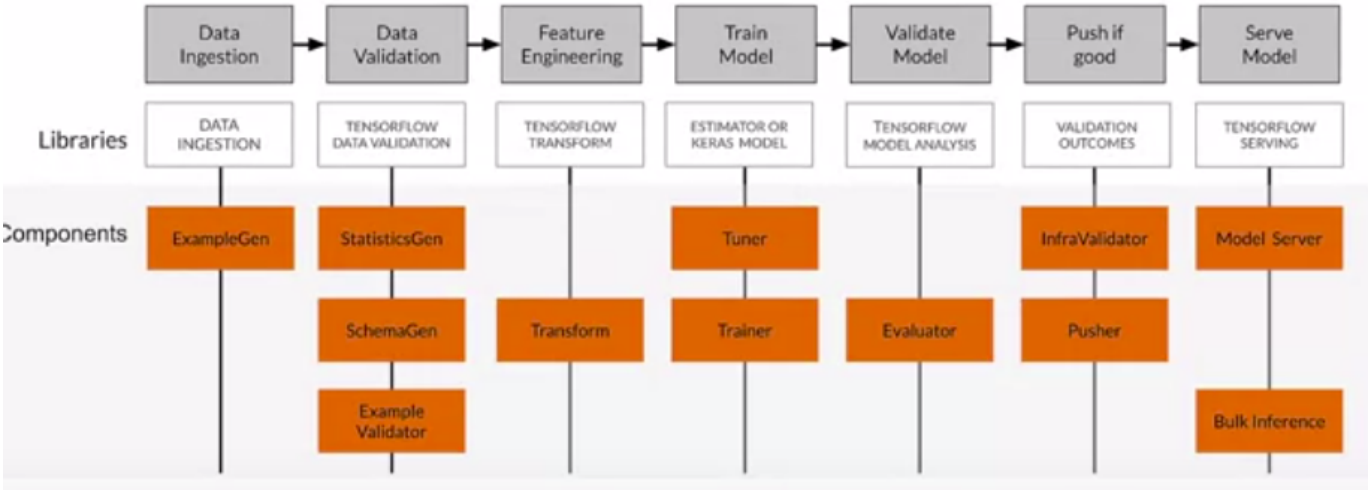
- Build integrated ML systems
- Continuously operate it in production
- Handle continuously changing data
- Optimize compute resource costs

ML pipelines



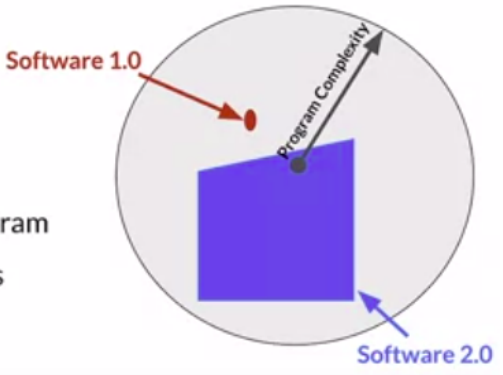
Infrastructure for
automating, monitoring, and maintaining
model training and deployment

TFX production components



ML: Data is a first class citizen

- Software 1.0
  - Explicit instructions to the computer
- Software 2.0
  - Specify some goal on the behavior of a program
  - Find solution using optimization techniques
  - Good data is key for success
  - Code in Software = Data in ML



Key points

- Understand your user, translate their needs into data problems
  - What kind of/how much data is available
  - What are the details and issues of your data
  - What are your predictive features
  - What are the labels you are tracking
  - What are your metrics