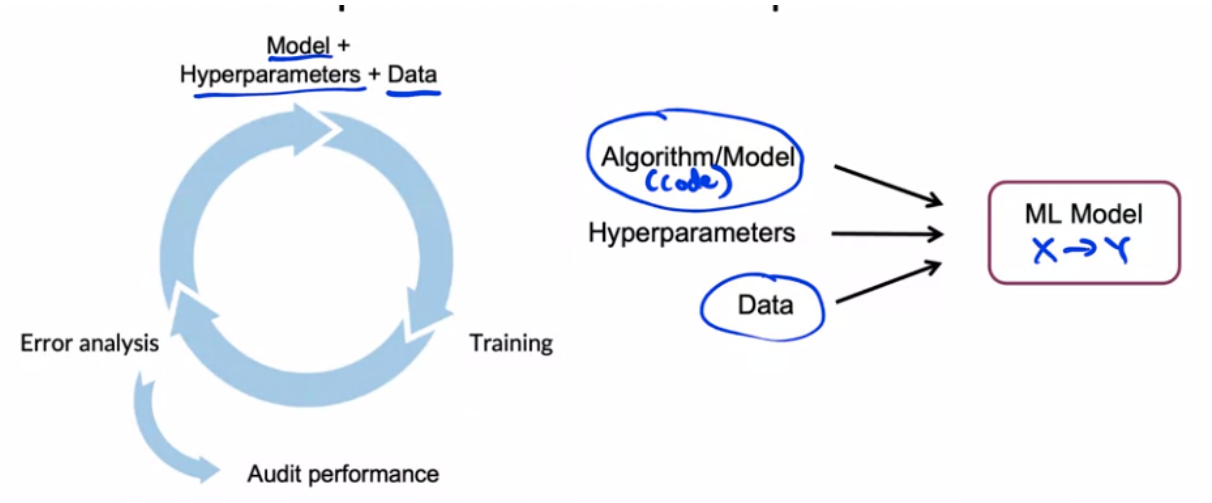
Weeks 2: Modeling

- **Model-Centric AI development:** This is more for researcher purposes.
- **Data-Centric AI development**: for practical purposes, this path is very important.

Key-challenges:

- AI system = CODE + Data

Model development is an iterative process:



Being able to go fast through the loop is very helpful! Each time in the loop it is necessary to analyze how to modify the model, data, or hyperparameters to upgrade our system.

Challenges in model development:

1. Doing well on the training set (usually measured by average training error)
2. Doing well on dev/test sets.
3. Doing well on business metrics/project goals.

Even if we achieve well on dev/test sets, that is not **GOOD** enough!

Why?

- Sometimes there are problems where the average test set's error is not measured equally, that's means there is a proportion of our data where the system needs to be more accurate.
- Performance on key slices of the dataset
    - Data discrimination or bias by ethnicity, gender, location, language, or other protected attributes. This is called a fairness algorithm!
- Rare classes - Skewed data distribution, this happens very often on medical's dataset. Accuracy in rare classes, when the classes are imbalanced so that cause for that specific class the sole accuracy does not hurt the whole accuracy.

Establish a baseline:

- Establish a baseline for all the cases. The best way is to analyze the human-level performance and compare it to our ML system results, with this we can know and spot where we need to improve.
- Literature search for state-of-the-art/open-source
- Quick-and-dirty implementation
- Performance of the older system

With Unstructured data and Structured data:

- As human is very good un Unstructured data such as images, audio, text, etc. We can use HLP (human-level performance) baseline as our ML system baseline.
- Structured data such as giant excel or database. Humans are not as good to look such as big data Structured data, so for this case, HLP is not a good baseline for our ML system.

These baselines help to indicates what might be possible. In some cases (such as HLP) it also gives a sense of what is an irreducible error/ Bayes error (The minimum error that it is possible to achieve).

Tips for Getting Started:

- Quick Literature search to see what's possible (courses, blogs, open-source projects).
- Find open-source implementations if available.
- A reasonable algorithm with good data will often outperform a great algorithm with no so good data.

Should we take into account deployment constraints when picking a model?
Yes, if the baseline is already established and the goal is to build and deploy!
No or not necessarily, if the purpose is to establish a baseline and determine what is possible and might be worth pursuing.
Sanity-check for code and algorithm:

- Try to overfit a small training dataset before training on a large one.
- Example with just one training example.
- Image segmentation: overfit in one image. (This lets you finds bugs faster)
- Image classification: use some batch of 100 or 10000 images, for these cases the training is very fast - overfit and look if the code has any bugs.

Error analysis and performance auditing:

- It is almost sure that your first model trained won't work!

Example: In speech recognition example

Use a spreadsheet to analyze validation samples with higher error or loss:

Example	Label	Prediction	Car Noise	People Noise
1	"Stir fried lettuce recipe"	"Stir fry lettuce recipe"	✓	
2	"Sweetened coffee"	"Swedish coffee"		✓
3	"Sail away song"	"Sell away song"		✓
4	"Let's catch up"	"Let's ketchup"	✓	✓

We can come with new tags different than *car noise* or *people noise*, this process helps you understand what is the source of the errors, it is possible to use automatic error analysis with MLOPS tools.

Error analysis is an iterative process:

Example tags in visual inspection:

- Image properties (blurry, dark background, light background, reflection, …)
- Other meta-data: phone model, factory, etc
- Specific class labels (scratch, dent, etc)

In product recommendation:

- User demographics
- product features/category

Useful metrics for each tag:

- What fraction of errors have that tag?
- Of all the data with that tag, what fraction is misclassified?
- What fraction of all the data has that tag?
- How much room for improvement is there on data with that tag? (Huma level performance or Bayes error)

It is better to brainstorm this tag while annotating the data because some of the previous questions need the whole data to be annotated with those tags to analyze the statistics of these tags.

Prioritizing what to work on:

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

Look to the % of data with those types of tags, and the human level performance difference, if the difference is high but the percentage of data is low, then focusing on improving that property won't be significant.

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

For categories you want to prioritize:

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

Skewed Datasets:

For medical diagnosis, manufacturing automatization, Speech recognition usually have very skewed datasets (Higher percent of positive examples than negative)

For this, we need to use **confusion matrix: Precision and Recall and 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%

Performance auditing:

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

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

3. Get business/product owner buy-in.

Data Iteration: (Data-centric development)

The quality of the data is paramount. Use tools to improve the data quality; this will allow multiple models to do well.

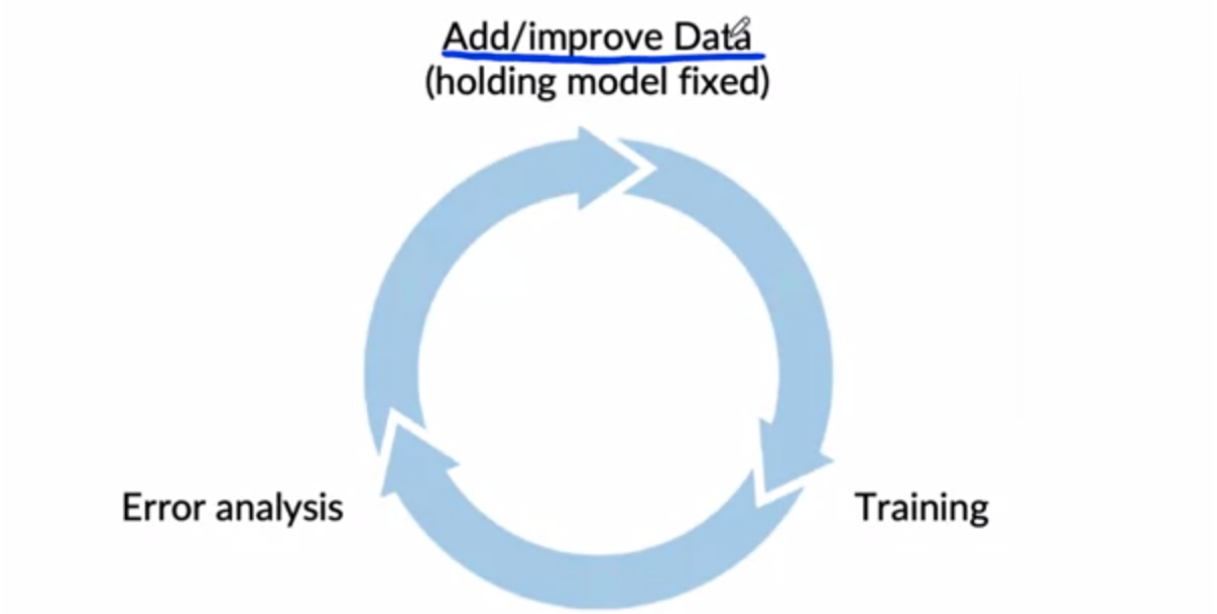
Data Augmentation:

Data augmentation in speech recognition:

- add Noice to input! What type of background noise? How loud will be the background noise relative to the speaker?

Goal:

- Create realistic examples that the algorithm does poorly on, but humans (or other baselines) do well on.
- 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 more data hurt?

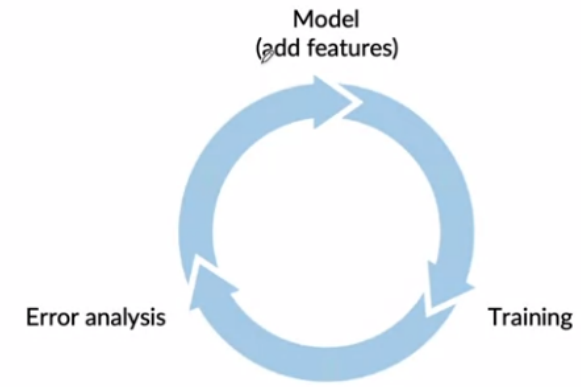
For unstructured data problems, if:

- The model is large (low bias).
- The mapping x->y is clear

then **adding data rarely hurts accuracy**

For structured data problems:

- Increase the number of features is a useful way to improve the algorithm performance



- Error analysis can be harder if there is no good baseline (such as HLP) to compare to.
- Error analysis, user feedback and benchmarking to competitors can all provide inspiration for features to add.

Experiment tracking:

- What to track?:
  - Algorithm/code versioning
  - Dataset used
  - Hyperparameters
  - Results
- Tracking Tools:
  - Manually Text files, Spreadsheets.
  - Systematically with Experiment tracking systems
    - MLflow
    - SageMaker studio
    - Weight&Biases
- Desirable features:
  - Information needed to replicate results, ideally with summary metrics/analysis
  - Resource monitoring, visualization, model error analysis.

Week 2 Optional References

If you wish to dive more deeply into the topics covered this week, feel free to check out these optional references. You won't have to read these to complete this week's practice quizzes.

Establishing a baseline

Error analysis

Experiment tracking

Papers

Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., … Anderljung, M. (n.d.). Toward trustworthy AI development: Mechanisms for supporting verifiable claims*. Retrieved May 7, 2021http://arxiv.org/abs/2004.07213v2

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2019). Deep double descent: Where bigger models and more data hurt. Retrieved from http://arxiv.org/abs/1912.02292