Step-by-Step Machine Learning Process

Cassie Kozyrkov (kozyr@google.com)

<u>What is Machine Learning?</u>

- Machine Learning (ML) is an approach to making many **decisions** that involves algorithmically **finding patterns** in old data and using these patterns to create models (recipes) for dealing correctly with **brand new situations.**
- In a nutshell: ML is all about finding and using patterns in data to perform new tasks.

<u>Jargon:</u>

- Instances: examples, observations. (rows)
- Labels: targets, ground truth. (correct answers)
- Features: information about the instances, variables, attributes. (columns)

<u>Check you need ML</u>

- If you can't even imagine what sort of decisions (labels) you'd like your ML system to make for you, stop. It might be too early for you to consider ML.
- Try imagining that you'd get 1 million free but distracted human workers who would all work on similar small tasks. What would you ask them to work on? What does good work look like? How would you know if they were slacking off? How would you choose between these million or those million free human workers? Imagine the work and imagine how you would assess its quality.

<u>Step-by-Step Machine Learning</u>

Step 1: Set Objective *(Decision Maker, Domain Expert)*

- Steps:
  - Write down outputs/labels.
  - Consider mistakes.
  - How would you score one mistake vs another.
  - Create business performance metric (BPM) by stitching together individual outcome scores for all of your individual outcomes together.
  - Look up some classic loss functions used for these types of outputs (e.g. loss functions for binary outputs, for text outputs, for numerical outputs, etc.)
  - Compare the business performance metric with the loss function.
  - Set minimum performance criteria to productionize and to launch.
- Thinking carefully about what success means and picking a metric that captures business performance is important! This is the decision-maker's responsibility and without it, the ML process is doomed.
- Imagine all the labels are made by an imperfect human worker instead of an ML system. Just focus on output.
- In ML, the proof of the pudding (model) is always in the eating (performance on new data). Always evaluate performance based on your business metric.

Step 2: Get Data *(Engineer, Domain Expert)*

- Your ML system is only as good as the data that went into it.
- Getting data involves lots of engineering effort.
- Just focus on getting IDs and a few inputs (“features” or “variables”) per ID. These won’t be the right ones, just a starting point for your analysts to work on in Step 4.

Step 3: Split Data *(Engineer)*

- Overfitting happens when you model noise instead of reality.
- If you don’t optimize for fit on totally fresh data, your models are no good to you.
- You need fresh data for checking performance!
- Split your data into:
  - Training dataset
  - Validation dataset
  - Test dataset
- Key Message: Your ML system is no good to you if it can’t deal with new data. It’s too easy to build a system that’s really good at old data but fails miserably on new. Make sure you avoid this by evaluating performance on fresh data.

Step 4: Explore (*Analyst, Domain Expert)*

- Plotting data is your secret weapon for machine learning.
- You’re only allowed to look in your training data!
- Don’t look in your validation and test datasets.
- Key Message: Your data are your most valuable resource. If you don’t explore your training dataset, you’re missing out on taking full advantage of it.

Step 5: Get Tools (Engineer)

- The math is in service of:
  1) Finding patterns (in old data).
  2) Assessing models (in new data).
- Unless you’re designing brand new algorithms, you can get away with:
  - Sufficient computer skills to use ML tools others have built
  - Sufficient statistical skills to evaluate model performance
- Key Message: Start with a list of available tools and aggressively eliminate everything that obviously won’t work, then just pick as many of the remaining tools and try them in parallel. Invest in the ability to try to run many algorithms in parallel.
- Don’t worry about picking the “right” algorithm. Worry about giving as many of them as possible a chance. The proof of the pudding is in the eating.

Step 6: Train *(Engineer, Analyst)*

- In training, your goal is to run lots of algorithms in parallel on your data and assess the models they produce on that same data.
- You’re making a shortlist of models that seem to work.
- Don’t worry about getting it right first time - it’ll take a few tries.
- Start simple and only build up the complexity if the simple solution doesn’t work.

Step 7: Tune and Debug *(Engineer, Analyst)*

- If you want the safest, most effective debugging strategy, then:
  - Run your algorithm (step 6) in some data.
  - Debug its performance by using it to label different data.
  - Since you’re not allowed to debug using validation or test data, you’re going to need to allocate a separate dataset for tuning/debugging. You’ll have a 4th dataset in play for debugging.
- You can create the tuning/debugging dataset on the fly by allocating some of the training data for this.

- If you have hyperparameters (numerical settings you must choose before running algorithm), use this data to tune them.

Step 8: Validate *(Analyst)*
- Validation is all about checking if your model succeeds on a new dataset.
- Validation protects you from blindly overfitting. It keeps you safe, don't skip it!
- Only view the final metric, not individual validation data points. Don't debug in your validation data.
- Repeatedly validating erodes your protection. That's why we have step 9 (testing).

Step 9: Test *(Decision Maker, Analyst)*
- Testing is the final frontier before you take your model live. This is where statistical rigor enters the picture.
- This is what the discipline of statistical inference is all about. Ask your statisticians for help.
- You only get one shot at this per test dataset.
- If testing fails, the only way to start again is to collect a pristine new test dataset.
- Never test on data that was involved in any way in training/tuning/debugging/testing.

Step 10: Build *(Engineer)*
- Your algorithm got you a favorite model (a model is just a recipe for turning inputs into outputs). The engineering team's job is to get this recipe into production.
- You can build it so it keeps itself updated automatically by building capabilities for retraining in production.
- Changing anything changes everything, so always test after a change.
- Don't forget to think about:
    - Retraining data, speed, and frequency.
    - Ability to restrict retraining data inputs and detect fleeting changes.
    - Logging bugs and changes to logging.
    - Tracking and safety nets for outliers.
    - Plans for when retesting fails.
- Policy layers (logic that checks the model output) are a very good idea. Build them before launching in production and make them easy to add case-by-case corrections to.

Step 11: Launch *(Analyst, Decision Maker)*
- You need to make sure the ML system is good enough for your business needs.
- Do an experiment to measure its impact and check that launching it at 100% is the right decision.
- Components of an experiment:
    - Hypothesis. (Performance of ML system good enough? This criterion was decided in Step 1.)
    - Different treatments. (ML system vs not ML system.)
    - Randomization to treatments. (Live traffic sent at random to ML system or old system.)

Step 12: Monitor and Maintain *(Reliability Engineer, Analyst)*
- Invest in:
    - Monitoring plan
    - Maintenance plan (and ensure there's headcount for carrying it out)
    - Tracking dashboards
    - Good documentation