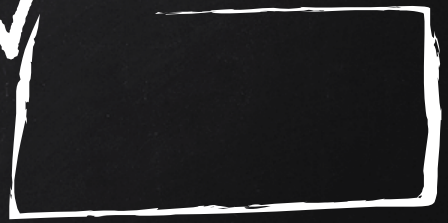
The End-to-End ML ~~Workflow~~



– Roger

- How well you apply machine learning theory to solve business problems.
- It isn't just about confronting a real-world problem like "How would you detect fraud on Amazon?" and then jumping to a technique like linear regression or random forests.
- Instead, it's about asking the right questions about the business goals and constraints that inform the machine learning system design.
- It's about walking through how you'd explore and clean the data and the features you would try to engineer.
- It's about picking the right model evaluation metrics and modeling techniques, contextualizing model performance in terms of business impact, mentioning model deployment strategies, and much, much more.

Clarify the problem and constraints

Some questions you can use to clarify the problem and the constraints include:

- What is the dependent variable we are trying to model? For example, if we are building a user churn model, what criteria are we using to define churn in the first place?
- How has the problem been approached in the past by the business? Is there a baseline performance we can compare against? How much do we need to beat this baseline for the project to be considered a success?
- Is ML even needed? Maybe a simple heuristics or a rule-based approach works well enough? Or perhaps a hybrid approach with humans in the loop would work best?
- Is this even legal or ethical to apply ML to this problem? Are there regulatory issues at play dictating what kinds of data or models you can use? For example, lending institutions cannot legally use some demographic variables like race.

- How do end users benefit from the solution, and how would they use the solution (as a standalone, or an integration with existing systems)?
- Is there a clear value add to the business from a successful solution? Are there any other stakeholders who would be affected?
- If an incorrect prediction is made, how will it impact the business? For example, a spam email making its way into your inbox isn't as problematic as a high-risk mortgage application accidentally being approved.
- Does ML need to solve the entire problem, end-to-end, or can smaller decoupled systems be made to solve sub-problems, whose output is then combined? For example, do you need to make a full self-driving algorithm, or separate smaller algorithms for environment perception, path planning, and vehicle control?

Once you've understood the business problem that you ML solution is trying to solve, you can clarify some of the technical requirements:

- What's the latency needed? For example, search autocomplete is useless if it takes predictions longer to load than it takes users to type out their full query. Does every part of the system need to be real time – while inference may need to be fast, can training be slow?
- Are there any throughput requirements? How many predictions do you need to serve every minute?
- Where is this model being deployed? Does the model need to fit on-device? If so, how big is too big to deploy? And how costly is deployment? For example, adding a high-end GPU to a car is feasible cost-wise, but adding one to a drone might not be.

Establish Metrics

- It's best to pick simple, observable, and attributable metrics that encapsulate solving the problem.
- Sometimes the business is only interested in optimizing their existing business KPIs (for example, the time to resolve a customer request). In that case, you need to be able to align your model performance metrics with solving the business problem. For example, for a customer support request classification model, a 90% mode accuracy means that 50% of the customer tickets that previously needed to be rerouted now end up in the right place, resulting in a 10% decrease in time to resolution.
- Recall that in the context of fraud detection, we've talked about accuracy, precision, recall, F-score, AUC, etc.

Understand your data sources

Data sources to consider:

- Can you acquire more data by crowdsourcing it via Amazon Mechanical Turk?
- Can you ask users for data as part of the user onboarding process?
- Can you buy second- and third-party datasets?
- Can you ethically scrape the data from online sources?
- Can you send your unlabeled internal data off to a labeling and annotation service?

Do you understand the data? Questions to consider:

- How fresh is the data? How often will the data be updated?
- Is there a data dictionary available? Have you talked to subject matter experts about it?
- How was the data collected? Was there any sampling, selection, or response bias?

Explore your data

- A good first step is to profile the columns at fist glance:
  - Which ones might be useful?
  - Which ones have practically no variance and thus wouldn't offer up any real predictive value?
  - Which columns look noisy?
  - Which ones have a lot of missing or odd values?
- Look at summary statistics like the mean, median, and
- Visualize your data:
  - For columns of interest, you want to visualize their distributions to understand their statistical properties like skewness and kurtosis.
  - Certain features (e.g., age, weight) may be better visualized with a histogram through binning.
  - Visualize the range of continuous variables and plot categorical variables
  - Correlation matrix

Clean your data

- One aspect of data munging is dropping irrelevant data or erroneously duplicated rows and columns.
- Handle incorrect values that don't match up with the supposed data schema.
- Handle missing value:
  - Imputing the missing values via basic methods such as column mean/median
  - Using a model or distribution to impute the missing
  - Dropping the rows with missing values (as a last resort)
- Deal with outliers:
  - Outliers may be due to issues in data collection, like manual data entry issues or logging hiccups. Or maybe they accurately reflect the actual data. Outliers can be removed outright, truncated, or left as is, depending on their source and the business implications of including them or not.
  - Note that outliers may be univariate or multivariate.

Feature engineering

Feature engineering is the art of presenting data to machine learning models in the best way possible.

- Feature selection
- Feature preprocessing
- Can you buy second- and third-party datasets?

The specific workflows for feature engineering depend on the type of data involved.

- Numeric feature
- Categorical feature
- Text feature

Model selection

Factors to consider when selecting a model include:

- Training & Prediction Speed: for example, linear regression is much quicker than neural networks for the same amount of data
- Budget: neural networks, for instance. Can be computationally intensive models to train
- Volume & Dimensionality of Data: for example, neural networks can handle large amounts of data and higher-dimensional data versus k-NN
- Categorical vs. Numerical Features: for example, linear regression cannot handle categorical variables directly, as they need to be one-hot encoded, versus trees (which can generally handle them directly)
- Explainability: choosing interpretable models like linear regression may be favorable to "black box" neural networks due to regulatory concerns or their ease of debugging

Model training & evaluation

Some talking points:

- Train-validation-test split
- Cross-validation
- Hyper-parameter tuning
- Feature importance
- Data imbalances
- Sampling
- …

DEPLOYMENT

The process of operationalizing the entire deployment process is referred to as "MLOps" – when DevOps meets ML. Generally, systems are deployed online, in batch , or as a hybrid of the two approaches.

- Online means latency is critical, and thus model predictions are made in real time.
- Batch means predictions are generated periodically and is helpful for cases where you don't need immediate results (most recommendation systems, for example) or require high throughput.

One deployment issue common to both batch and online ML systems is model degradation.

- Feature drift
- Training–serving skew
- How often to retrain a model, what events might trigger a model refresh, and how much new data to use versus how much to rely on historical data
- How to add logging to monitor and catch model degradation

ITERATE

- Model deployment isn't the end of the machine learning workflow.
- Knowing how to iterate on your system can be learning via error analysis.
- By continuously seek ways for your system to improve, you can generate increasing amounts of business value with your deployed models

Reference

Ace The Data Science Interview by Kevin Huo and Nick Singh

Machine Learning System Design Interview by Ali Aminian and Alex Xu