Data Science!

By Eric Schles

Difference Between machine learning and engineering

- Science is about falsifiability
- Software Engineering is an art not a science
  - Readability over correctness
  - Convention over creativity
  - You do the same thing again and again, aka web dev, mobile dev
- Data Science and Computer Science are similar which makes this difference subtle and non-obvious, especially to students

Problem Definition

- The way you define your problem defines your solution
  - Falsifiability is essential
- Your problem should indicate what kind of modeling you will do
  - Time series?
  - Classification?
  - Regression?
  - NLP?
  - Image processing?
- Your problem should be solvable with standard tools or it should be clear the problem is novel, from the definition

An example of good problem definition:
https://yorko.github.io/2023/scaling-laws-near-dups/?trk=feed-detail_comments-list_comment-text

Defining Acceptance Criteria

- Defines what's good enough -
  - An MSE of 0.5, an F1 score of 80%, etc
    - The measures you use to define acceptance matter, a lot
    - But the more important thing is how this affects business value
      - An MSE of 1.0 would lead to a loss in business value of 1 billion dollars a year, because of the cost of an error, therefore an MSE of 0.5 is required to cover the costs of the system (the stakes are almost never this high)
  - It's important that acceptance criteria allows for success or failure based on the particulars of the system
  - Never build a system that costs more to build and maintain then it makes or saves over the medium or long term

Acceptance Criteria case study - real world

- Hopkins maintains a system that pulls information from an upstream system
- The upstream system has data that isn’t always correct, so we monitor the volume of records passing into our database tables, using time series forecasting
- We need to measure acceptance of the models based on:
  - Precision (type 1 error)
  - Recall (type 2 error)
  - F1 score (type 1 & type 2 error)
  - This is because missing errors (type 2) is as important as flagging non-errors (type 1)

How to model

- Train, test and validation data sets are important (if at all possible)
  - Train with hyperparameter tuning
  - Validate hyperparameter tuning with test data
  - Final validation inspection should only happen with the ‘best model’ with the best hyperparameters
  - Repeat this for at least a few random seeds - don’t bet on one train test split!
  - https://github.com/EricSchles/randomizer_ml && https://github.com/EricSchles/randomness_experiments/blob/master/scikit-learn-experiments-breast_cancer.ipynb

How to model

- Assume your data will drift
  - Once your model goes to production, make sure you are monitoring your model through acceptance criteria
  - Your data is almost definitely going to drift, have a plan ready for when that happens
- Don’t leak information between train, test, validation
  - Do separate preprocessing on train, test and validation
  - Don’t rebalance test or validation if you rebalance train
  - Act as though you won’t have test or validation data, so make sure you can’t cheat by adjusting
- Always check for simpson’s paradox
  - Look for subgroups in your data that reverse the relationship with your overall population relationship

Many more gotchas: https://www.youtube.com/watch?v=BDEBF62iZx0&ab_channel=PyData

Dealing with unbalanced data

- https://github.com/EricSchles/datascience_book/blob/master/9/unbalanced%20data.ipynb
-

TIME SERIES MODELS

https://github.com/EricSchles/datascience_book/blob/master/6/Introduction%20to%20timeseries%20modeling.ipynb

The importance of hypothesis testing and modeling

We don’t build models only as engineering end products, we build them to learn about and better understand our systems. Models are a forcing function for understanding your system in detail