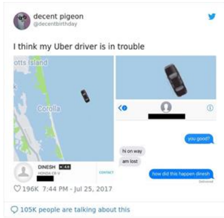
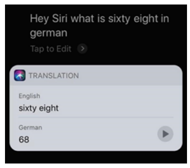
Data Analysis

Model Evaluation





Facebook translates 'good morning' into 'attack them', leading to arrest

Palestinian man questioned by Israeli police after embarrassing mistranslation of caption under photo of him leaning against bulldozer

0 People Like You

Model evaluation

- Offline evaluation: before deployed
  - Performance metrics (we covered these already!)
  - Error analysis
  - Bias/variance analysis
  - Robustness metrics
- Online evaluation: after deployed If train the model, it is perfect, then let's Release to Prod

Test in production. We don't this discuss this for now

I trained a classifier with **80% accuracy**. What's next? 😕📊

Is that good? Depends

Add more data? 📈📁
Train the algorithm for longer? ⏳🔄
Use a bigger model? 🏗️💡
Add regularization? 🎯⚖️
Add new features? 🧩🔍

Performance baselines

- Numbers by themselves mean little
- Task: binary classification, 90% POSITIVE, 10% NEGATIVE
- F1 score: 0.90

Is the model good or bad?

Model selection: baselines then we compare the simple models

- **Random baseline** Specially in Classifier
  - Predict at random:
    - uniform
    - following label distribution
- **Zero rule baseline**
  - Always predict the most common class
- **Simple heuristics**
  - e.g.: classify tweets based on whether they contain links to unreliable sources
- **Human baseline** Image Rec, text Classif
  - what's human-level performance?
- **Existing solutions**
  - E.g. out-of-the-box models etc.

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% (990,000) negative (label = 0)
  - 1% (10,000) positive (label = 1)

	Accuracy	F1
Random [uniform]	0.5	0.02
Random [label distribution]	0.98	0.01
Most common [preds = [0] * n]	?	?
Simple heuristics	?	?
Human expert	?	?
3rd party API	?	?

Error analysis

- Error analysis systematically identifies the most common model errors.
  1. Collect a random subset of misclassified dev set examples.
  2. Manually examine them; identify the most common *categories*.
  3. Count \% of points affected by each error categories.
- You should prioritize the most common error categories.

- Error analysis involves classifying errors into *categories*.
  - A category is a systematic error made by the model.
  - Categories can come from intuition, but ideally they are constructed by manually examining the data.

Error analysis: An example

- Suppose you just trained a new image classifier.
  - After looking at the first 20-30 errors, you realize they are all due to images being either *blurry*, *flipped*, or *mislabeled*.
  - These become the error categories.

	Blurry	Flipped	Mis-labeled
Img 1	X	X	
Img 2		X	
Img 3			X
...	...	...	...
Total	20%	50%	30%

- Real-world data is often messy and labels are not always correct
  - **Scenario 1**: 10% error, of which 0.4% is due to labeling and 9.6% to other causes. No need to relabel.
  - **Scenario 2**: 2% error, of which 0.4% is due to labeling and 1.6% to other causes. Important to relabel!

Error analysis but on what?

Would you do an error analysis on...:

A. The training set?
B. The validation set? → The Correct One ✓
C. The test set? ✗ wrong
D. The yellow unicorns from previous lectures?

Bias/variance analysis

- **Overfitting**: A very expressive model fits the training dataset perfectly.
  The model makes incorrect predictions on the validation set (high variance)
- **Underfitting**: The model is too simple to fit the data well
  It is not accurate on training data and is not accurate on new data (high bias)

- Both are problematic:
  - High bias: Our model is not able to extract useful info from the data
  - High variance: Whatever the model picked up, it does not generalize to new data

Quantifying bias and variance

$$\text{dev error} = \underbrace{(\text{dev error} - \text{train error})}_{\text{variance}} + \underbrace{\text{train error}}_{\text{bias}}$$

- Bias is simply the training error
- Variance is the extra error on the validation set

- We use these observations to diagnose bias/variance in practice
  - High training error → High bias
  - High validation error → High variance

OK, Over- or Under-fit?

- Training error 15%
  Validation error 16%
  Human error 2%
  Under or OK
- Training error 4%
  Validation error 12%
  Human error 2%
  Overfit
- Training error 2.5%
  Validation error 3%
  Human error 2%
  OK

A. OK
B. Overfit
C. Underfit
D. Unicorns love Fridays

Addressing variance Overfitting

- Give the model more data 📂📈 Ideal Solution
- Add regularization 🎯⚖️ can increase ↓ Variance
- Drop or change features 🔄🧩
- Reduce model size 📉🤏

Addressing bias

- Increase the expressivity (complexity) of the model 🐻 + 1 Scale
- Add or change features 🔧
- Reduce regularization ⚖️🔒
- Adjust model architecture 🏗️🔄

Robustness methods

1. Perturbation Tests
2. Invariance Tests
3. Directional Expectation Tests
4. Model Calibration
5. Confidence Measurement
6. Slice-based Evaluation

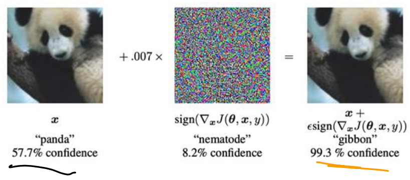
Perturbation tests

- Problem: users input might contain noise, making it different from test data
  - Examples:
    - Speech recognition: background noise
    - Object detection: different lighting
    - Text inputs: typos, intentional misspelling (e.g. looooooooong)
  - Model does well on test set, but fails in production

differ on how People are Gonna Use it

Perturbation tests

- Motivation: users input might contain noise, making it different from test data
- **Idea**: randomly add small noise to test data to see how much outputs change
- The more sensitive the model is to noise:
  - The harder it is to maintain
  - The more vulnerable the model is to adversarial attacks



gets Confused

Perturbation tests For Tabular Data

- Motivation: users input might contain noise, making it different from test data
- **Idea**: randomly add small noise to test data to see how much outputs change

- For tabular data:
  1. Select feature(s) to check Perturb
  2. Define the perturbation strategy (e.g. random noise, value shifting/swapping, masking, outliers)
  3. Apply perturbation and check performance

seeing Changes the Performance

If small changes cause model's performance to fluctuate, you might want to make model more robust:
- Add noise to training data
- Add more training data
- Choose another model

Invariance tests (also for fairness)

- Motivation: some input changes shouldn't lead to changes in outputs
  - Changing race/gender info shouldn't change predicted approval outcome
  - Changing name shouldn't affect resume screening results

The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

Disparity in home lending costs minorities millions, researchers find (CBS News, 2019)

Invariance tests

- Motivation: some input changes shouldn't lead to changes in outputs
- **Idea**: keep certain features the same, but randomly change values of sensitive features

If changing sensitive features can change model's outputs, there might be biases!

Directional expectation tests

- Motivation: some changes to inputs should cause predictable changes in outputs
  - E.g. when predicting housing prices:
    - Increasing lot size shouldn't decrease the predicted price
    - Decreasing square footage shouldn't increase the predicted price
- **Idea:** keep most features the same, but change certain features to see if outputs change predictably

If increasing lot size consistently reduces the predicted price, you might want to investigate why!

Model calibration

- If you predict team A wins in A vs. B match with 60% probability:
  - In 100 A vs. B match, A should win 60% of the time!

We have the expectation that in 100 games between the 2

Model calibration: binary case

Among all samples predicted POSITIVE with probability 80%, 80% of them should be POSITIVE

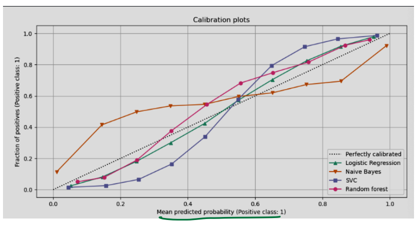


Image from Probability calibration (sklearn)

Model calibration: recsys

- Recommend movies to a user who watches 70% comedy, 30% action
- What happens if you recommend most likely watched movies?

Movie title	Watch probability
Comedy 1	0.8
Comedy 2	0.73
Comedy 3	0.68
Comedy 4	0.67
Action 1	0.29
Action 2	0.2
Science fiction	0.04

Need to calibrate recommendations to include 70% comedy, 30% action

Confidence measurement

- Usefulness threshold for each individual prediction
- Uncertain predictions can cause annoyance & catastrophic consequences

- How to measure the confidence level of each prediction?
  - Probability distribution (e.g. logistic regression or softmax)
  - Margin of Confidence (difference between the 2 probabilities)
  - More advanced methods (e.g. conformal prediction)
- What to do with predictions below the confidence threshold?
  - Skip
  - Ask for more information
  - Loop in humans

Slice-based evaluation

Gap Dee

Different performance on different slices

- Classes
  - Might perform worse on minority classes

- Subgroups
  - Gender
  - Location
  - Time of using the app
  - etc.

Same performance on different slices with different cost

- User churn prediction (i.e. when a customer stops engaging with a service)
  - Paying users are more critical
- Predicting adverse drug reactions
  - Patients with underlying conditions are more critical

⚠ Focusing on improving only overall metrics might hurt performance on subgroups ⚠

Slice-based evaluation: example

- Majority group: 90%
- Minority group: 10%

Coarse-grained evaluation can hide:
- model biases
- potential for improvement

	Majority accuracy	Minority accuracy	Overall accuracy
Model A	98%	80%	96.2%
Model B	95%	95%	95%

Simpson's paradox

- Models A and B to predict whether a customer will buy your product
- A performs better than B overall
- B performs better than A on both female & male customers

Simpson's paradox

	Treatment 1	Treatment 2
Group A	**93% (81/87)**	87% (234/270)
Group B	**73% (192/263)**	69% (55/80)
Overall	78% (273/350)	**83% (289/350)**

Numbers from a kidney stone treatment study. (Charig et al., 1986)

Slice-based evaluation

- Evaluate your model on different slices
  - E.g. when working with website traffic data, slice data among:
    - gender
    - mobile vs. desktop
    - browser
    - location
- Check for consistency over time
  - E.g. evaluate your model on data slices from each day

- Improve model's performance both overall and on critical data
- Help avoid biases
- Even when you don't think slices matter, slicing can:
  - give you confidence on your model (to convince your boss)
  - might reveal non-ML problems

How to identify slices?

- Heuristics
  - Might require subject matter expertise (consult domain experts)
- Error analysis
  - Patterns among misclassified samples
- Slice finder (i.e. use a model)
  - Exhaustive/beam search
  - Clustering
  - ...

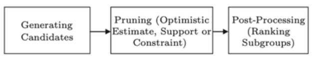


Fig.1. Methodology for subgroup discovery.

Slice finder: Automated data slicing for model validation (Chung et al., 2019)
Subgroup Discovery Algorithms: A Survey and Empirical Evaluation (Sumyea Helal, 2016)