

A Practical Guide to Machine Learning

Hongsup Shin

Arm Research

Dec 5, 2018

About me

- Academic background
 - Masters: Evolutionary biology (insect vision)
 - PhD: Computational neuroscience (human vision)
- Industry experience
 - Oil and gas consulting start-up
 - Arm
- Current role at Arm
 - ML researcher / Data scientist
 - Machine learning on hardware verification

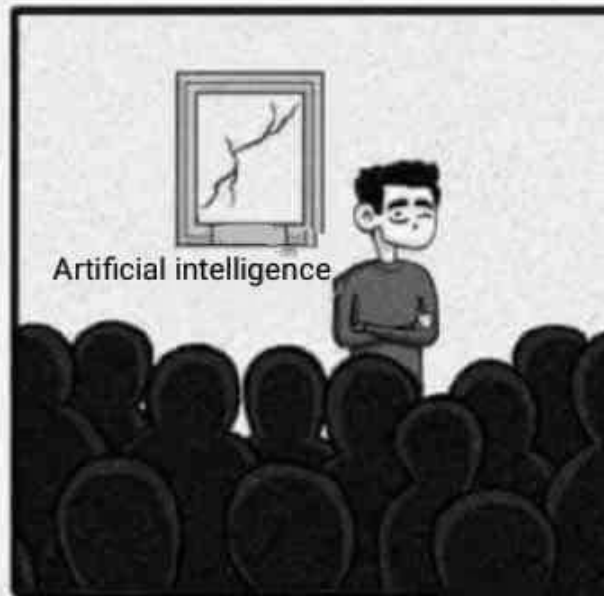
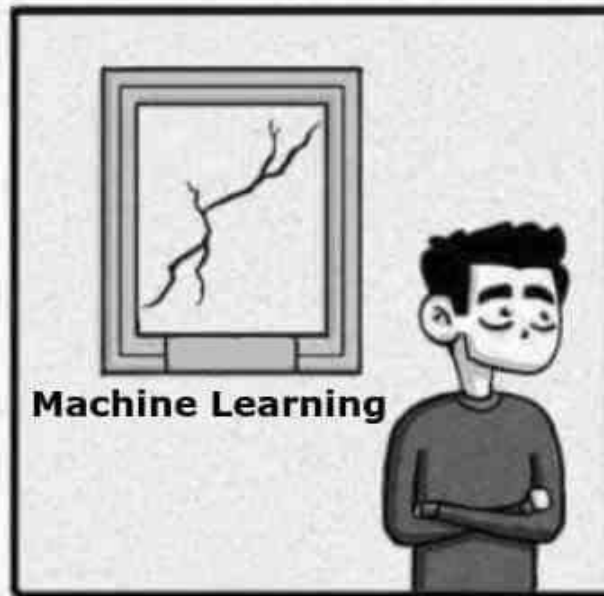
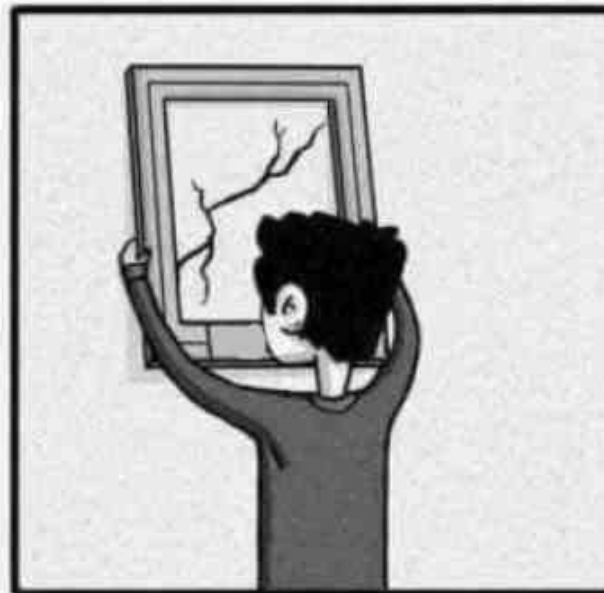
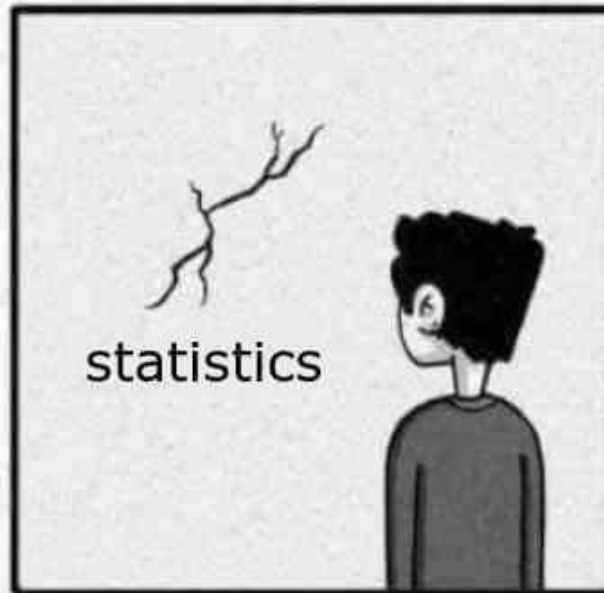
Today

We will NOT talk about...

- Computer architecture
- Details of ML algorithms
- Deep learning (DL) and its tools

But we will talk about...

- How a ML project works:
example and guide
 - How to examine data
 - What questions to ask & what to check
 - Communication
- The dark side of ML
- How to study ML



Why ML needs careful approach

- The ML hype
 - Many don't know whether they need it or how to properly use it
 - ML is code-light, concept-heavy: easy to implement, difficult to do it right
- ML is about efficiency = Automating human task/decision-making
 - With humans removed from decision making due to ML, flaws can be overlooked and populated in a massive level
 - Do it right and be responsible
- ML engineering is complex
 - Careful and methodical approach

ML algorithms

- Essentially, learning a pattern (optimization)
- **Supervised**: there is a right answer
 - Two types of datasets: input and output
 - Prediction: with a new set of given input, predict the possible output
 - Classification (yes/no choice) : “Will I get my PhD in 5 years?”
 - Regression (continuous): “How many years will it take for me to get a PhD?”
- **Unsupervised**: there is no clear answer
 - One type of dataset
 - Clustering: “group the students based on X”
 - **More difficult because it’s hard to validate**

Real-world ML applications

Using satellite imagery to generate awareness and funds for refugees

Automatically Counting Tents in Refugee Camps

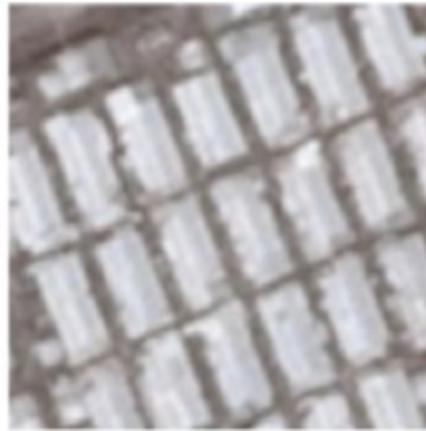
Satellite imagery showing tents in South Sudan refugee camp: Algorithm automatically counts images of tents

Here are a couple of predicted images.

Pred : 28 | Act : 28



Pred: 24 | Act 29



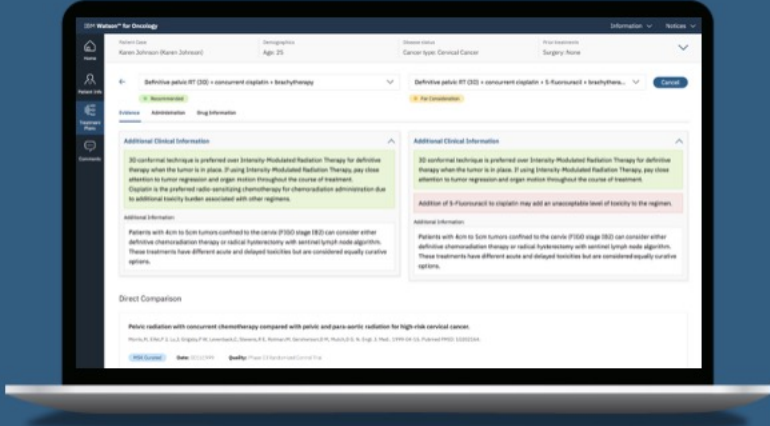
- Save staff and volunteer time
- Allow UNHCR to better serve refugees by quickly estimating refugee camp populations
- Identifying locations to fill gaps in service (e.g., providing communal latrines, water points, etc.)

Personalized medicine

IBM Watson for Oncology

Watson for Oncology helps physicians quickly identify key information in a patient's medical record, surface relevant evidence and explore treatment options.

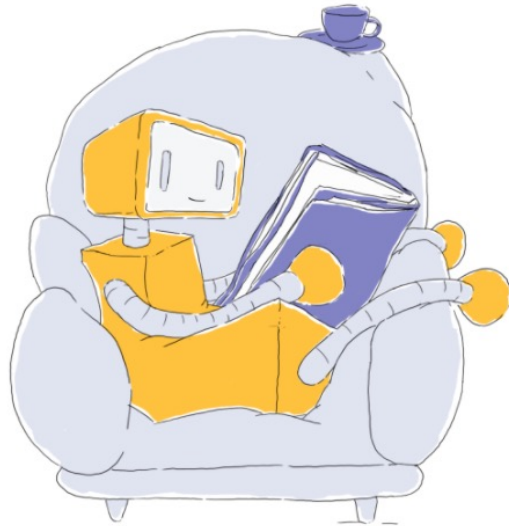
[Watch the video \(05:07\)](#)[Contact Us](#)



The image shows a laptop displaying the IBM Watson for Oncology interface. The screen displays a patient profile for Karen Johnson, a 55-year-old female with Cervical Cancer. The interface includes a sidebar with navigation options like Home, Patient Info, and Reports. The main content area shows treatment recommendations, clinical trial information, and a direct comparison of treatment options. The interface is designed to help physicians quickly identify key information in a patient's medical record and explore treatment options.

- Personalized medicine based on individual health data paired with predictive analytics
- Hot research area and closely related to better disease assessment.
- Presently ruled by supervised learning: physicians to select from more limited sets of diagnoses or estimate patient risk based on symptoms and genetic information.

Cognitive behavioral therapy



**Created by experts.
Backed by research.**

Woebot was created by leading experts in clinical psychology
and has demonstrated ability to make people happier.

[Learn more about our research](#)

woebot.io

- Automated conversational agent (Woebot) for behavioral therapy and mood tracking
- “What’s going on in your world right now?”, “How are you feeling?”, etc.
- A decision tree with natural language processing technique

ML engineering: guide and examples

ML engineering cycle

- **Problem statement (consultation)**
- **Envision and implement the full pipeline (product)**
- **Data collection and preprocessing**
- **Model building and training (evaluation and selection)**
- **Deployment**
- **(Retraining)**

- **Repeat!**

Example: bug detection in hardware design verification

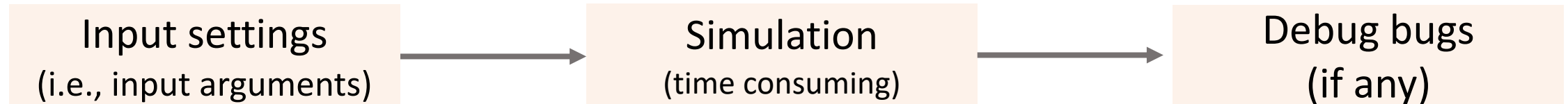
- "Does this proposed design do what is intended?"
- A *design* is made of hardware description language
- Simulation-based design testing: run simulations to provide stimulus to test each line in the code
- Random testing: provide random stimuli to verify random design parts (pass or fail due to bugs)
- Engineers must find and fix bugs until bug-free
- The faster you find them, the better!

```
module test( input wire clk,
             input wire [1:0] i,
             output wire [2:0] o );

  covergroup cg @(posedge clk);
    coverpoint o {
      option.name = "test/o";
    }
  endgroup

  cg cg_inst = new;
  wire temp;
  assign temp = (i[0] == 1'b1) ? 1'b0 : 1'b1;

  always@(*) begin
    case( i[1:0] )
      2'b00: o[2:0] = {temp, 2'b11};
      2'b01: o[2:0] = {temp, 2'b10};
      2'b10: o[2:0] = {temp, 2'b01};
      2'b11: o[2:0] = {temp, 2'b00};
      default: o[2:0] = {3{1'bx}};
    endcase
  end
endmodule
```

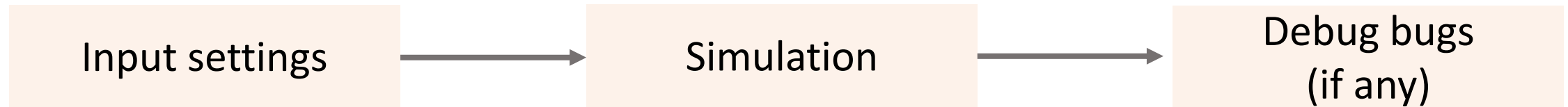


Challenges in design verification

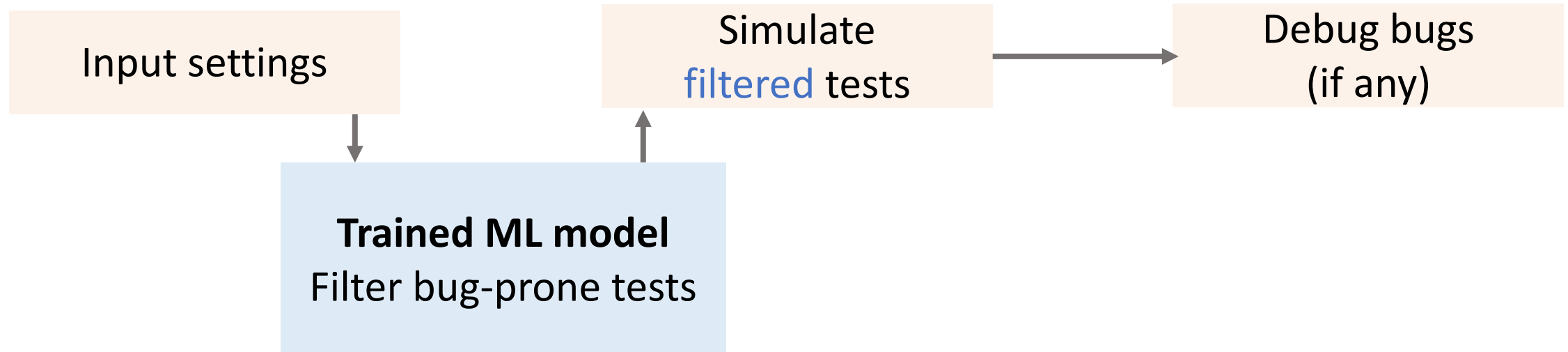
- Massive volume of tests due to combinatorial complexity in design
- Inefficient and poorly guided randomized testing
 - Redundant tests
 - As we are left with fewer bugs, tests should be guided to them but with random testing, they are not.
- Manual intervention (gut feeling): neither accountable nor reproducible
- Instead of running random tests, can we select tests that are likely to create bugs?

“Can ML tell us which tests are more bug-prone so that we don’t have run all tests?”

Current workflow



Proposed workflow with ML



Run basic sanity checks

- Data quality: garbage in, garbage out (next slide)
- Baseline performance: no machine learning (random testing and manual intervention)
- Implementation (it's a collaborative engineering process!)
 - How, who, when: command-line tool, prototype by research then maintenance by production, 2-3 months
 - New feature vs. improvement: not a replacement but a complementary tool

Data matter A LOT



Data sanity checks: garbage in, garbage out

Data is more important than using fancy models

Data infrastructure Database, tabular files, unstructured	No database but tabular files (csv) Relatively clean (parsed)
Collection cost	Cheap new data generation
Batch vs. live stream	Batch
Data types Check the no. of unique values (e.g., sign of an integer)	Substantial amount of binary columns

Data sanity checks: garbage in, garbage out

Input

	Setting 1	Setting 2	Setting 3	Setting 4	...
Test 1	0.5	1	'high'	1	...
Test 2	0.2	1	'med'	0	...
Test 3	0.8	0	'low'	1	...
...

Output

	Bug? (Yes/No)
Test 1	0
Test 2	0
Test 3	1
...	...

Quantity

Roughly 100 data points per feature

50k rows (tests)

400 columns (input setting features)

Imbalance of the output data

High (99% pass, 1% bug)

Your first model

- Deprioritizes machine learning gains, avoid getting distracted
- Define objectives clearly and consider **practical significance**
 - “Even if it is true that value X is 0.1% more than value Y, does it matter?”
- Choose a simple and observable metric
- Starting with an interpretable model makes debugging easier: linear regression, logistic regression (**no neural nets!**)
- **For bug detection...**
 - Supervised vs. unsupervised?
 - Regression vs. classification?

Test infrastructure independently from ML

- If you are building a ML **tool**, your infrastructure must be solid.
- You only need a placeholder model for testing infrastructure.
- Test getting data in and out of the trained model.
- Check future dataset gets same treatment as training data.
- Make sure that the model in your training environment gives the same score as the model in your serving environment

Come up with a robust validation scheme

- Split the data into training, validation and testing set.
- Training: train your model
- Validation: tune your models and to check **overfitting**
- Testing: test if your final model is general enough for unseen future data **(make sure you never touch the testing data until the very end)**
- Ideally your metric should be similar between all three.
- How to split
 - equal proportion, training more than others, or cross-validation
 - **Make sure your testing set is not too small.**

The Cardinal Sin of ML: Overfitting

- Your model is tuned so well to the training data that **it fails to generalize to new unseen data** (memorizing all answers)
- **Overfitted model is useless and DANGEROUS!**
- Training error < validation error
 - Usual case (matter of degree). Not always bad.
 - Too much then overfitting
- Training error > validation error
 - Unusual case
 - Bug or *underfitting*

Choose your model performance metric

True	Pass (0)	True negative	False positive
	bug (1)	False negative	True positive
		Pass (0)	bug (1)
		Predicted	

Classification metric

- Accuracy: $(TP + TN) / All$
- Recall: $TP / (TP + FN)$:
false negative matters
- Precision: $TP / (TP + FP)$:
false positive matters

Metrics for imbalanced data

True	Pass (0)	TN: 90	FP: 0
	bug (1)	FN: 10	TP: 0
		Pass (0)	bug (1)
		Predicted	

- Accuracy: $(TP + TN) / \text{All} = 90\%$

- Recall: $TP / (TP + FN) = 0$

- Precision: $TP / (TP + FP) = 0$

Trained
model



“Every test will pass so don’t worry about it and let all tests go!”

- By predicting everything as pass, we can get high accuracy easily.
- Accuracy is not the best for imbalanced data.

Metrics for imbalanced data

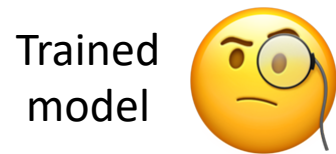
True	Pass (0)	TN: 89	FP: 1
	bug (1)	FN: 6	TP: 4
		Pass (0)	bug (1)
		Predicted	

- Accuracy: $(TP + TN) / \text{All} = 93\%$

- Recall: $TP / (TP + FN) = 40\%$

Recall or precision?

- Precision: $TP / (TP + FP) = 80\%$



“These 5 tests are likely bugs. You should keep them.”

Consider the cost of false predictions

- False positive: the true is a pass but model says it's a bug
 - Model keeps this test and we can run it
 - Consequence: waste of resource
- False negative: the true is bug but model says it's a pass
 - Model lets this bug-test escape
 - Consequence: bug-test not arrested
- The cost of false negative is higher than that of false positive
 - Recall ($TP / (TP + FN)$) might be our metric of choice.

Choose your algorithm wisely

- First vs. subsequent launch
- Interpretable vs. prediction-focused
 - Interpretable: we want to know which feature does what
 - Prediction-focused: we don't care how things work but want good prediction
- Computational resource, training time, data size, retraining frequency

Logistic regression as our first pick

- One of data scientists' go-to “off-the-shelf” *classification* model
- Linear model: straightforward and interpretable
- Fast training and prediction time
- Very few *hyperparameters*



Model implementation

- Tool: Python scikit-learn package
- Validation scheme
 - 25% of the data saved as testing
 - The rest (75%) is used for 3-fold cross-validation
- Model
 - Logistic regression (hyperparameters not tuned)
 - Make sure random state is fixed for debugging
 - Class weight applied (higher cost for misclassifying bugs)

Results (testing set)

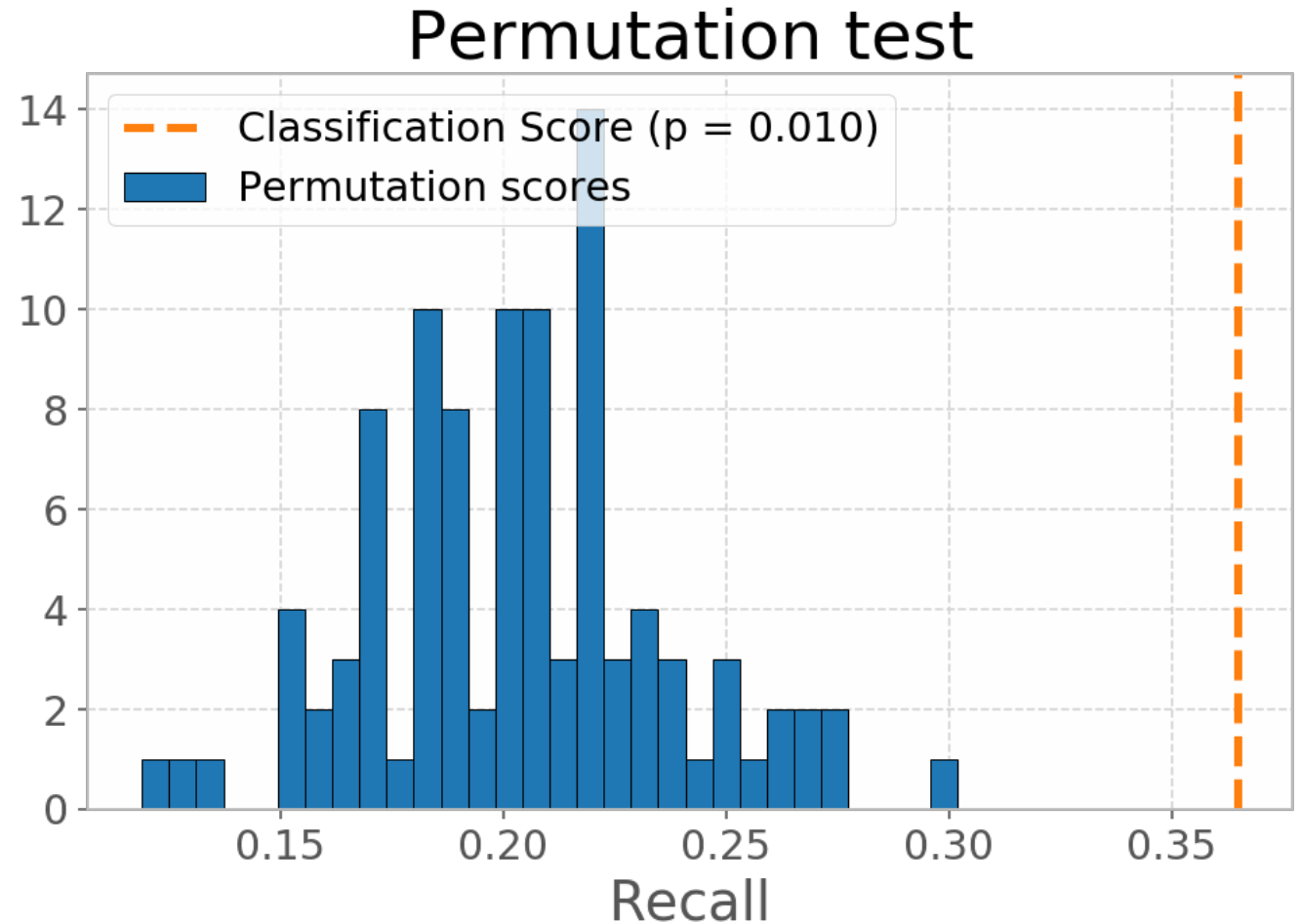
True	Pass	Bug	
	9747	2632	
Bug	58	65	
Predicted			
		Pass	Bug

- The trained model says to run only 20% of all tests.
- The trained model can capture 50% of all bugs.
- x2.5 times more efficient than random testing

Sanity check: Is my model better than random?

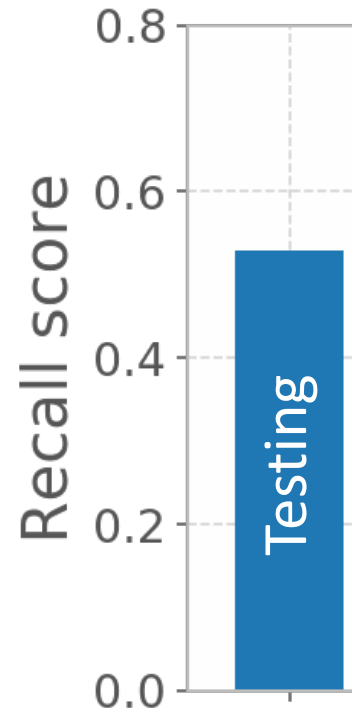
Permutation test

1. Randomly shuffle your output data and build a model.
2. Repeat N times and check whether your trained model performance is better than the random simulations.



Test the robustness of your model

- Computing performance metric on new incoming data
- Consistent results across multiple new datasets



Sending out the trained model to real world

- Track metric and raise a flag when model performance $<$ threshold
- Model performance may change if there is...
 - a bug in your pipeline
 - data drift
- Keep improving the model...
 - with newly collected data
 - by tuning the model
 - by engineering features
- Consider retraining
- There will be a lot of launches!

Advanced troubleshooting: error analysis

- When your ML model makes an error in prediction, investigate it.
 - ML system knows it got wrong and would like to fix if given the opportunity.
 - If you give the model a feature that allows it to fix the error, the model will try to use it.
- Once you have examples that the model got wrong, look for trends that are outside your current feature set. (try to find a pattern in errors)

Storytelling and documentation

- You don't work alone. You often need to explain results to non-experts.
- Data analysis starts with questions, not data or a technique.
 - Analysis without a question will end up aimless.
 - Avoid the trap of finding some favorite technique and then only finding the parts of problems that this technique works on.
- Acknowledge and document every decision you made.

Communication

- Use visualizations as much as possible
- Ratios should have clear numerator and denominators
- Provide the context and a full picture, not just a number
- Be both skeptic and champion
 - “What other data could I gather to show how awesome this is?”
 - “What could I find that would invalidate this?”
- Expect and accept ignorance and mistakes
 - Credibility is the key social value for any data scientists and ML practitioners.

The dark side of ML

ML as double-edged sword

- Not the ML per se but human bias and malicious intention going into ML
- Relationship between technology decisions, policy and how business can exploit the vulnerability in unforeseen consequences and tech-ignorant policy
- It's better to intervene early on before tech moves too much forward; joint effort from regulators, advocates and journalists

INSIDE THE TWO YEARS THAT SHOOK FACEBOOK—AND THE WORLD

One employee compared Zuckerberg to Lennie in *Of Mice and Men*—a man with no understanding of his own strength.

Online discrimination and privacy

- Potential for discrimination in online targeted ads
 - Facebook ad-personalization violates the fair housing act
 - Facebook disabled the targeting but is it still possible to create targeted ads?
- Discrimination in online personalization
 - Google's gender-biased algorithms: image search, google translate, etc.
 - Google allows specific jobs targeted for specific gender
- Anonymous data dump has privacy issues
 - Muslim drivers were identified in NYC cab data

Gender stereotyping in Google Translate

DETECT LANGUAGE HUNGARIAN **ENGLISH** SPANISH ▼ ↔ ENGLISH **HUNGARIAN** SPANISH ▼

He is a doctor,
He is a nurse.
She is an engineer.

→ ×

Ő orvos,
Ő ápolónő.
Ő egy mérnök.

50/5000

🔊 📄 ✎ 🔗

DETECT LANGUAGE **HUNGARIAN** ENGLISH SPANISH ▼ ↔ **ENGLISH** HUNGARIAN SPANISH ▼

Ő orvos,
Ő ápolónő.
Ő egy mérnök.

→ ×

🌟 Did you mean: Ő orvos, **ÉS** ápolónő. Ő egy mérnök.

33/5000

🔊 📄 ✎ 🔗

Interpretability and explainability

- EU's general data protection regulation (GDPR): in Europe, data protection is a fundamental right.
- Explainability in automated decision is mandatory; general ban on solely automated processing
- Accountability in automated decision making

Fairness in computer vision



Joy Buolamwini, MIT Media Lab

- Most face recognition algorithms misidentify the gender of dark-skinned population: misidentify female as male
- 130M adults is in a law enforcement face recognition network
- Creating a new benchmark dataset to correct the bias and testing existing algorithms: Face++, Microsoft, IBM (different accuracy by gender and race)

Attacks against machine learning

- Adversarial inputs: craft inputs that can be reliably misclassified to evade detection
- Data poisoning attacks (model skewing): attacker attempts to pollute training data to manipulate the decision boundary in her favor
- Model stealing: “steal” (i.e., duplicate) models or recover training data membership

(Bursztein, 2018)

Adversarial inputs: disinformation (fake news)

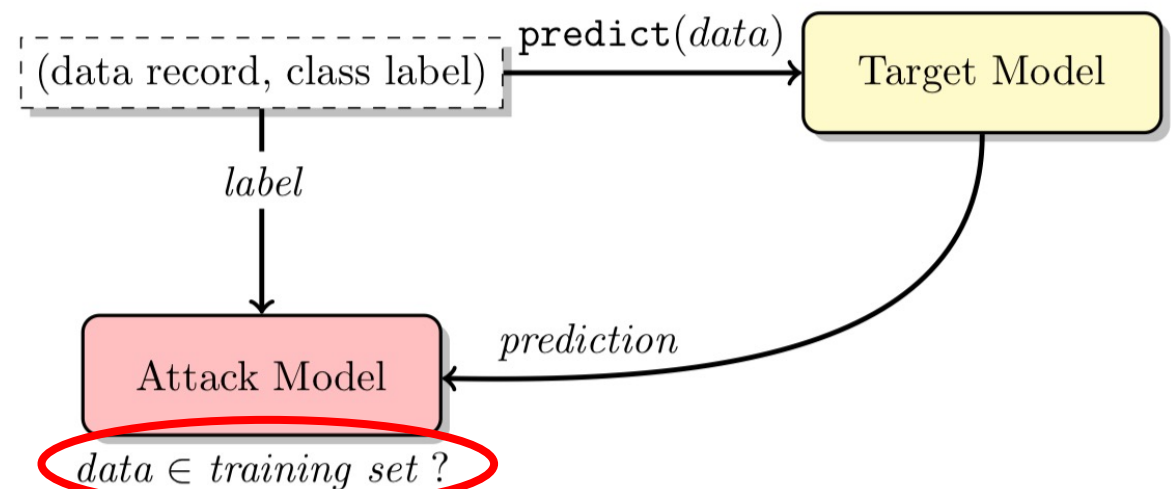
Model stealing attacks

- Training-data extraction attacks (model inversion attacks)
 - Fredrikson et al. (2015)



Figure 1: An image recovered using a new model inversion attack (left) and a training set image of the victim (right). The attacker is given only the person's name and access to a facial recognition system that returns a class confidence score.

- Membership inference attacks
 - Shokri et al. (2016)



Responsible and accountable ML

- Existing human bias and prejudice can be reproduced in a massive scale through machine learning.
- ML practitioners and data scientists should be aware of potential data privacy issues.
- To avoid these problems, we need accountable ML.
- To build a good ML product, interdisciplinary effort is required.

ML resources

- Online lectures
 - Coursera: Andrew Ng's ML and deep learning specialization
 - Udacity: Intro to Machine Learning
 - Google: Machine learning crash course
- Textbooks
 - Practical + Python examples: Python Machine Learning (Raschka), 2nd Edition
 - Deep learning: Deep Learning with Python (Chollet, creator of Keras)
 - In-depth: Introduction to Statistical Learning (aka ISLR), Pattern Recognition and Machine Learning (Bishop)
- Online articles
 - Google: "Rules of Machine Learning"
 - Data Science Weekly Newsletter
 - Blog posts (be critical)
- Do your own project

How to start ML projects/career

Academic research	<ul style="list-style-type: none">• Probably the easiest (since you are grad students)• Need to find the right use case
Lecture assignments	<ul style="list-style-type: none">• Many online lectures have exercises.• Difficult to be creative
Toy projects	<ul style="list-style-type: none">• Need to find the right data and right questions (data.world)• Doesn't have to solve world hunger
Volunteering	<ul style="list-style-type: none">• Opportunity for collaborative experience (need to provide skills)• Sometimes lacks leadership (but you can be a leader!)
Competition	<ul style="list-style-type: none">• \$\$ (if you win) and can learn a lot in a short time period• Prediction-focused ML
Internship	<ul style="list-style-type: none">• \$\$ (if they pay you), easier to get a full-time job• Need to have some skills already but less intense interviews
Bootcamps	<ul style="list-style-type: none">• Easier to get a full-time job afterwards• Usually very expensive (>10k)
Master's degree	<ul style="list-style-type: none">• Very expensive• Somewhat overkill

Summary

- Machine learning is an engineering process.
- Understand the problem, check the data, consider your product pipeline before making any modeling decisions.
- Good communication skills can solve many problems easily.
- ML engineering is complicated; be cautious and methodical.

Closing remarks

- ML now affect many aspects of human activities.
- To build something useful with ML, treat it like a scientific engineering process: it requires rigor and efficiency.
- ML is like any technology; double-edged sword. As an engineer/scientist/researcher, be responsible.
- Have fun with ML but do it right and be more informed on how your work affects others.

Thank you

hongsup.s@gmail.com

hongsup.shin@arm.com

Retraining

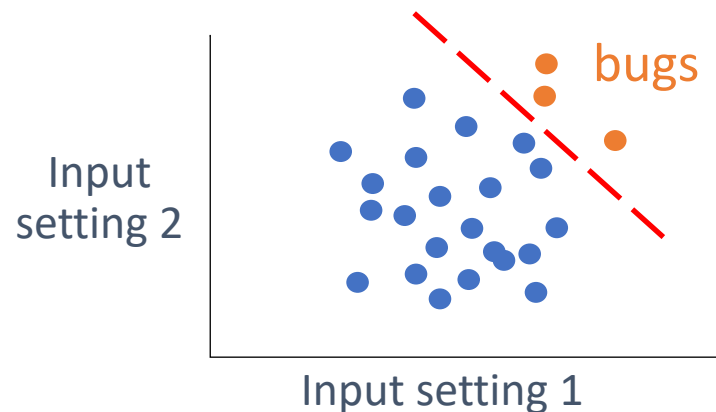
- Many disturbances to underlying data happen as our systems evolve over time.
- Just because a particular day or set of days is an outlier does not mean you should discard it. Use the data as a hook to find a causal reason.
- Know the freshness requirements of your system.
 - Example: feature columns are added or removed in the new data
- Retraining is costly. Consult with clients and domain experts first.

Consider your data when choosing models

- High imbalance in data (500 bugs out of 50,000 tests)
- ML models need to detect bug-prone tests

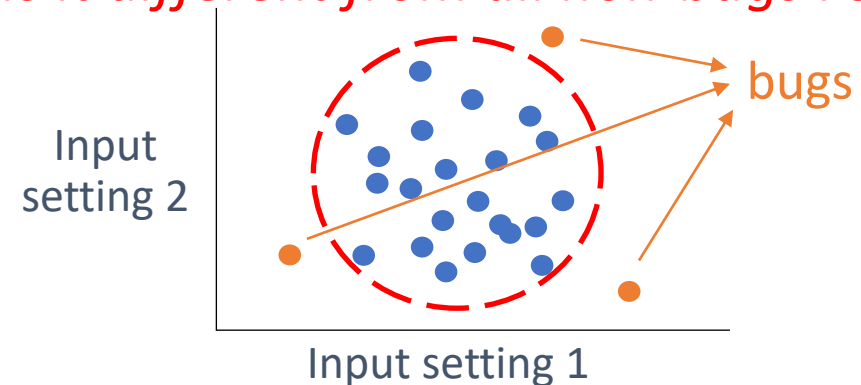
Supervised

- Require passes and bugs
- Learn decision boundary between bug and pass
- *Is it similar to the bug I saw?*



Unsupervised

- Require only passes
- Learn a boundary around passes and flag new data as bug if they are outside
- *Is it different from all non-bugs I saw?*



Choose a model: pros and cons

Supervised

- Need to have bugs in training data
- Straightforward and fast training
- **Cannot explore new types of bugs**
(not suitable for exploring novel bugs)
- Can validate model performance

Unsupervised

- No bugs required
- Complex and slower training
- Potentially explore new bug types
- **If there is no bug, impossible to validate the results**

- We do have bugs in our dataset.
- We can use them to train and validate our model.
- **Classification: pass vs. bug**