Machine Learning in Production

Model Correctness and Accuracy

Administrativa

- Waitlist now closed, teams assigned
- HW1 due tonight, unless you joined late, then Sep 14
- Each team will receive links with details on how to access a virtual machine for the team project soon (by Monday)
- See Canvas for other cloud resources (optional)

Learning Goals

- Select a suitable metric to evaluate prediction accuracy of a model and to compare multiple models
- Select a suitable baseline when evaluating model accuracy
- Know and avoid common pitfalls in evaluating model accuracy
- Explain how software testing differs from measuring prediction accuracy of a model

Model Quality

First Part: Measuring Prediction Accuracy

- the data scientist's perspective

Second Part: What is Correctness Anyway?

- the role and lack of specifications, validation vs verification

Third Part: Learning from Software Testing

- unit testing, test case curation, invariants, simulation (next lecture)

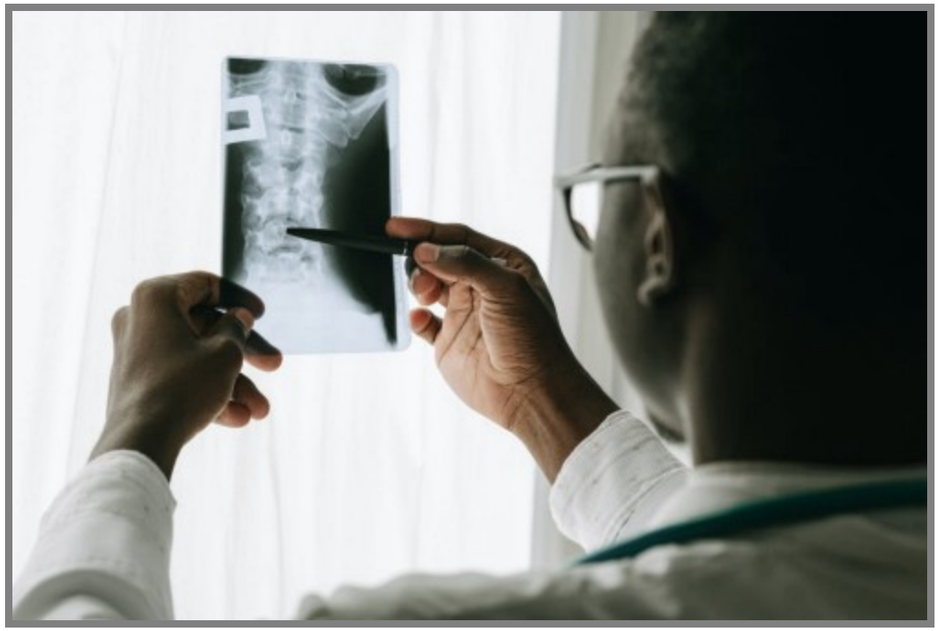
Later: Testing in Production

- monitoring, A/B testing, canary releases (in 2 weeks)

Case Study & Scope

(Model Quality, Prediction Accuracy)

Case Study: Cancer Prognosis

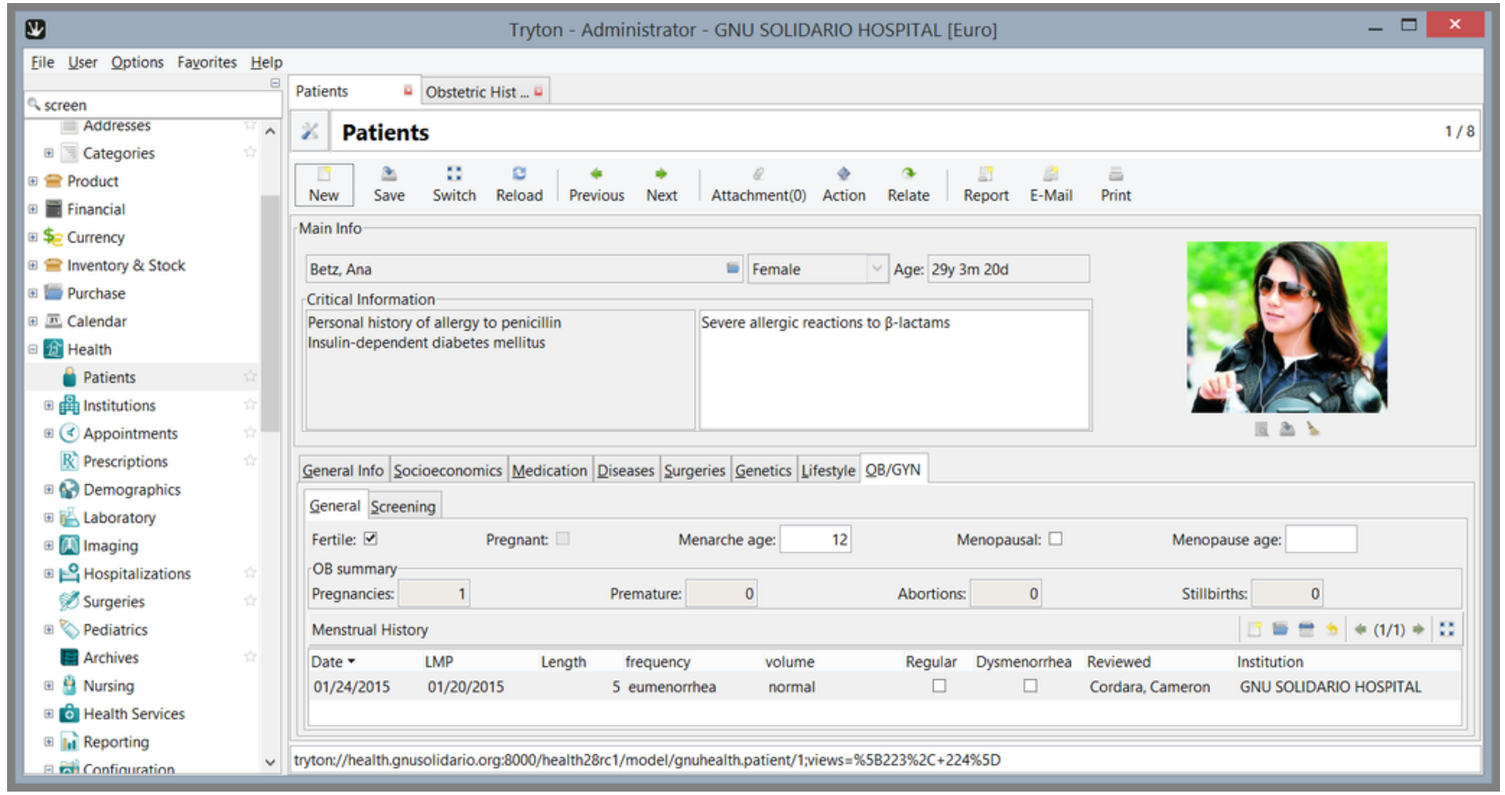


We should stop training radiologists now. It's just completely obvious that within five years, deep learning is going to do better than radiologists. -- Geoffrey Hinton, 2016

Speaker notes

Application to be used in hospitals to screen for cancer, both as routine preventative measure and in cases of specific suspicions. Supposed to work together with physicians, not replace.

The Model is Part of a System in an Environment



Scope: Model Quality, not ML Algorithm/Data/System Quality

Todays focus is on the quality of the produced *model*, not the algorithm used to learn the model or the data used to train the model or the product around the model

- ➤ assuming *Decision Tree Algorithm* and feature extraction are correctly implemented (according to specification), is the model learned from data any good?

Focus on measuring quality, not debugging the source of quality problems

Out of Scope Today: System-Level Considerations

- Models used by radiologists, humans in the loop
- Radiologists are specialists who do not directly see patients
- Radiologists may not trust model, but are also overworked
- Radiologist must explain findings
- Patient can see findings before physician (CURES act)

Out of Scope Today: Model Qualities beyond Accuracy

But many *model qualities* matters when building a system beyond just *prediction accuracy*, e.g.:

- Model size
- Inference latency
- Learning latency
- User interaction model
- Ability to incrementally learn
- Explainability
- Robustness

Today and Next Lecture

Narrow focus on prediction accuracy of the model

That's difficult enough for now.

More on system vs model goals and other model qualities later

On Terminology

Model: $\overline{X} \rightarrow Y$

Training/validation/test data: sets of $(\overline{X}, Y)$ pairs indicating desired outcomes for select inputs

Performance: In machine learning, "performance" typically refers to accuracy: "this model performs better" = it produces more accurate results

Be aware of ambiguity across communities (see also: performance in arts, job performance, company performance, performance test (bar exam) in law, software/hardware/network performance)

- When speaking of "**time**", be explicit: "learning time", "inference latency", ...
- When speaking of model **accuracy** use "prediction accuracy", ...

Part 1:

Measuring Prediction Accuracy for Classification Tasks

(The Data Scientists Toolbox)

Confusion/Error Matrix

	Actually Grade 5 Cancer	Actually Grade 3 Cancer	Actually Benign
Model predicts Grade 5 Cancer	**10**	6	2
Model predicts Grade 3 Cancer	3	**24**	10
Model predicts Benign	5	22	**82**

$$accuracy = \frac{correct\ predictions}{all\ predictions}$$

Example's accuracy = $\frac{10+24+82}{10+6+2+3+24+10+5+22+82} = .707$

```
def accuracy(model, xs, ys):
  count = length(xs)
  countCorrect = 0
  for i in 1..count:
    predicted = model(xs[i])
    if predicted == ys[i]:
      countCorrect += 1
  return countCorrect / count
```

Typical Questions

Compare two models (same or different implementation/learning technology) for the same task:

- Which one supports the system goals better?
- Which one makes fewer important mistakes?
- Which one is easier to operate?
- Which one is better overall?
- Is either one good enough?

Is 99% Accuracy good?

Is 99% Accuracy good?

➤ depends on problem; can be excellent, good, mediocre, terrible

10% accuracy can be good on some tasks (information retrieval)

Always compare to a base rate!

Reduction in error = $$\frac{(1-accuracy_{\text{baseline}})-(1-accuracy_{f})}{1-accuracy_{\text{baseline}}}$$

- from 99.9% to 99.99% accuracy = 90% reduction in error
- from 50% to 75% accuracy = 50% reduction in error

Baselines?

Suitable baselines for cancer prognosis? For audit risk prediction?

Speaker notes

Many forms of baseline possible, many obvious: Random, all true, all false, repeat last observation, simple heuristics, simpler model

Consider the Baseline Probability

Predicting unlikely events -- 1 in 2000 has cancer (stats)

Random predictor

	Cancer	No c.
Cancer pred.	3	4998
No cancer pred.	2	4997

.5 accuracy

Never cancer predictor

	Cancer	No c.
Cancer pred.	0	0
No cancer pred.	5	9995

.999 accuracy

See also Bayesian statistics

Measures

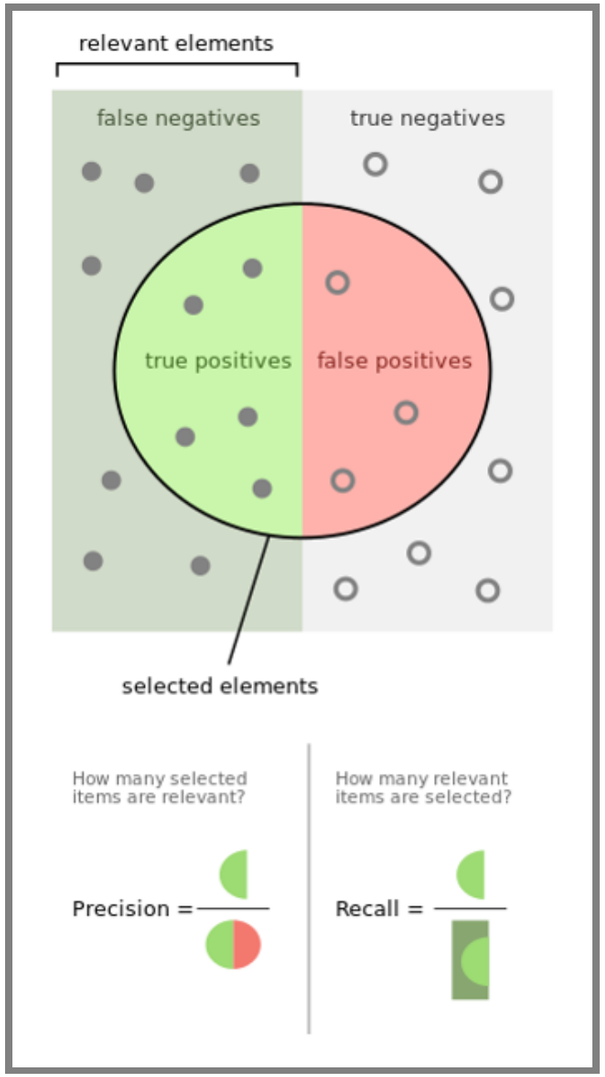
Measuring success of correct classifications (or missing results):

- Recall = TP/(TP+FN)
  - aka true positive rate, hit rate, sensitivity; *higher is better*
- False negative rate = FN/(TP+FN) = 1 - recall
  - aka miss rate; *lower is better*

Measuring rate of false classifications (or noise):

- Precision = TP/(TP+FP)
  - aka positive predictive value; *higher is better*
- False positive rate = FP/(FP+TN)
  - aka fall-out; *lower is better*

Combined measure (harmonic mean): F1 score = $2\frac{recall*precision}{recall+precision}$



(CC BY-SA 4.0 by Walber)

False positives and false negatives equally bad?

- Recognizing cancer
- Suggesting products to buy on e-commerce site
- Identifying human trafficking at the border
- Predicting high demand for ride sharing services
- Predicting recidivism chance
- Approving loan applications

No answer vs wrong answer?

(This requires understanding the larger system!)

Extreme Classifiers

- Identifies every instance as negative (e.g., no cancer):
  - 0% recall (finds none of the cancer cases)
  - 100% false negative rate (misses all actual cancer cases)
  - undefined precision (no false predictions, but no predictions at all)
  - 0% false positive rate (never reports false cancer warnings)
- Identifies every instance as positive (e.g., has cancer):
  - 100% recall (finds all instances of cancer)
  - 0% false negative rate (does not miss any cancer cases)
  - low precision (also reports cancer for all noncancer cases)
  - 100% false positive rate (all noncancer cases reported as warnings)

Consider the Baseline Probability

Predicting unlikely events -- 1 in 2000 has cancer (stats)

Random predictor

	Cancer	No c.
Cancer pred.	3	4998
No cancer pred.	2	4997

.5 accuracy, .6 recall, 0.001 precision

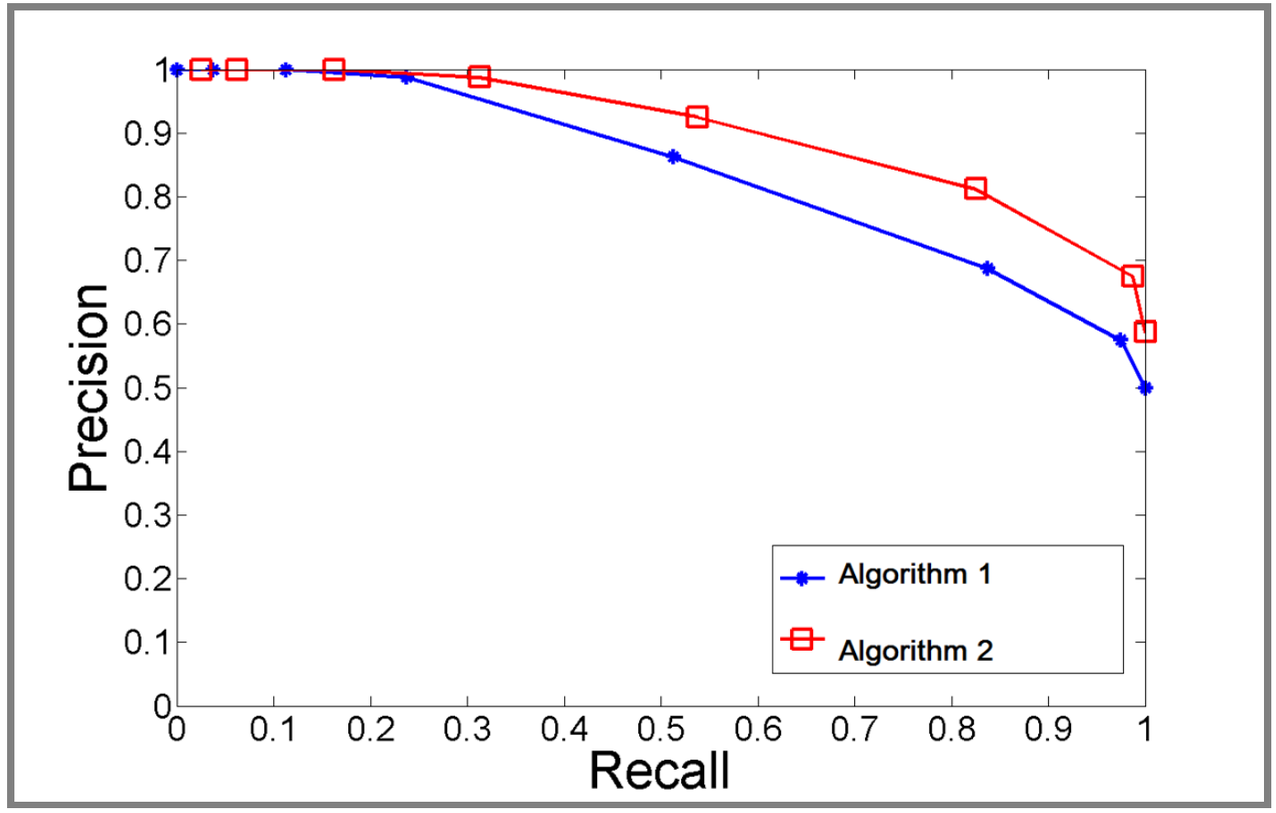
Never cancer predictor

	Cancer	No c.
Cancer pred.	0	0
No cancer pred.	5	9995

.999 accuracy, 0 recall, .999 precision

Area Under the Curve

Turning numeric prediction into classification with threshold ("operating point")



Speaker notes

The plot shows the recall precision/tradeoff at different thresholds (the thresholds are not shown explicitly). Curves closer to the top-right corner are better considering all possible thresholds. Typically, the area under the curve is measured to have a single number for comparison.

More Accuracy Measures for Classification Problems

- Lift
- Break even point
- F1 measure, etc
- Log loss (for class probabilities)
- Cohen's kappa, Gini coefficient (improvement over random)

Many Measures Beyond Classification

Regression:

- Mean Squared Error (MSE)
- Mean Absolute Percentage Error (MAPE)
- R^2 = percentage of variance explained by model
- ...

Rankings:

- Mean Average Precision in first K results (MAP@K)
- Mean Reciprocal Rank (MRR) (average rank for first correct prediction)
- Coverage (percentage of items ever recommended)
- Personalization (how similar predictions are for different users/queries)
- ...

Natural language processing:

- Translation and summarization -> comparing sequences (e.g ngrams) to human results with specialized metrics, e.g. BLEU and ROUGE
- Modeling text -> how well its probabilities match actual text, e.g., likelyhoold or perplexity

Always Compare Against Baselines!

Measuring Generalization

The Legend of the Failed Tank Detector

Speaker notes

Widely shared story, authenticity not clear: AI research team tried to train image recognition to identify tanks hidden in forests, trained on images of tanks in forests and images of same or similar forests without tanks. The model could clearly separate the learned pictures, but would perform poorly on other pictures.

Turns out the pictures with tanks were taken on a sunny day whereas the other pictures were taken on a cloudy day. The model picked up on the brightness of the picture rather than the presence of a tank, which worked great for the training set, but did not generalize.

Pictures: https://pixabay.com/photos/lost-places-panzer-wreck-metal-3907364/, https://pixabay.com/photos/forest-dark-woods-trail-path-1031022/

Overfitting in Cancer Prognosis?

Separate Training and Validation Data

Always test for generalization on *unseen* validation data (independently sampled, typically from the same distriution)

Accuracy on training data (or similar measure) used during learning to find model parameters

```
train_xs, train_ys, valid_xs, valid_ys = split(all_xs, all_ys)
model = learn(train_xs, train_ys)

accuracy_train = accuracy(model, train_xs, train_ys)
accuracy_valid = accuracy(model, valid_xs, valid_ys)
```

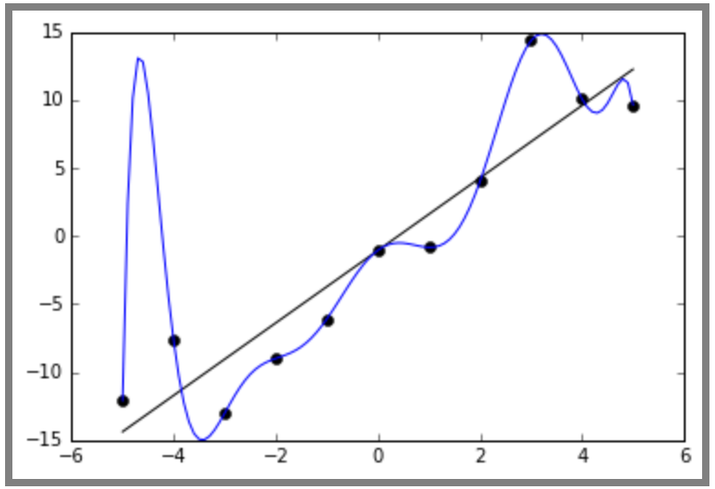
$accuracy\_train >> accuracy\_valid$ = sign of overfitting

Overfitting/Underfitting

Overfitting: Model learned exactly for the input data, but does not generalize to unseen data (e.g., exact memorization)

Underfitting: Model makes very general observations but poorly fits to data (e.g., brightness in picture)

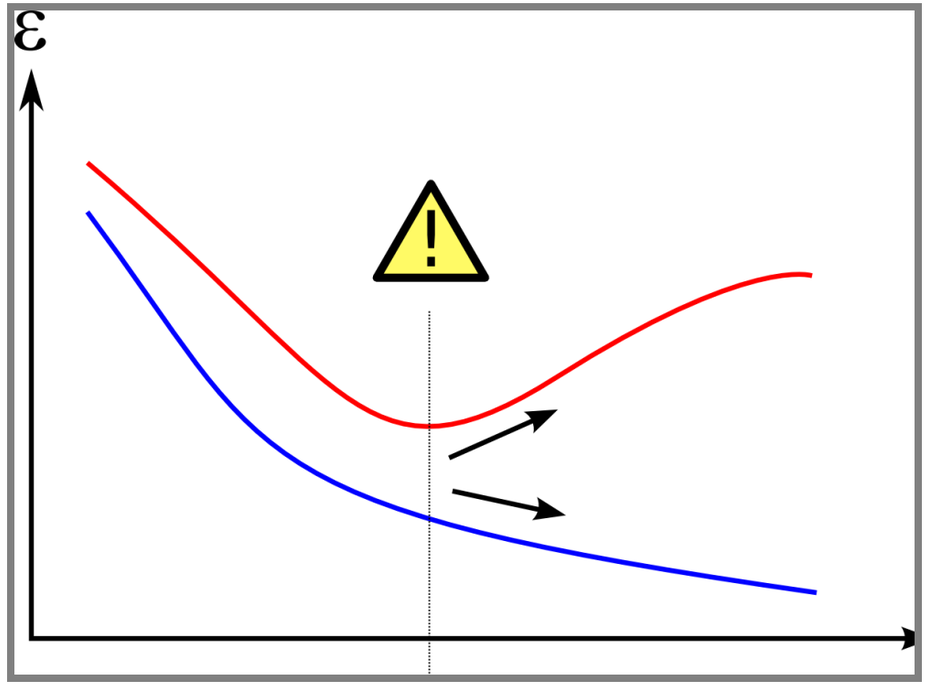
Typically adjust degrees of freedom during model learning to balance between overfitting and underfitting: can better learn the training data with more freedom (more complex models); but with too much freedom, will memorize details of the training data rather than generalizing



(CC SA 4.0 by Ghiles)

Detecting Overfitting

Change hyperparameter to detect training accuracy (blue)/validation accuracy (red) at different degrees of freedom



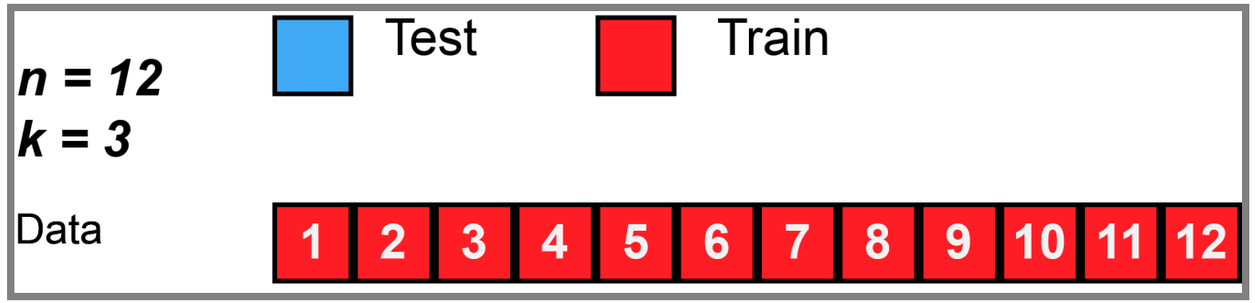
Speaker notes

Overfitting is recognizable when performance of the evaluation set decreases.

Demo: Show how trees at different depth first improve accuracy on both sets and at some point reduce validation accuracy with small improvements in training accuracy

Crossvalidation

- Motivation
  - Evaluate accuracy on different training and validation splits
  - Evaluate with small amounts of validation data
- Method: Repeated partitioning of data into train and validation data, train and evaluate model on each partition, average results
- Many split strategies, including
  - leave-one-out: evaluate on each datapoint using all other data for training
  - k-fold: k equal-sized partitions, evaluate on each training on others
  - repeated random sub-sampling (Monte Carlo)



Production Data -- The Ultimate Unseen Validation Data

more in a later lecture

Changelog
@changelog · **Follow**

“Don’t worry, our users will notify us if there’s a problem”

2:03 PM · Jun 8, 2019

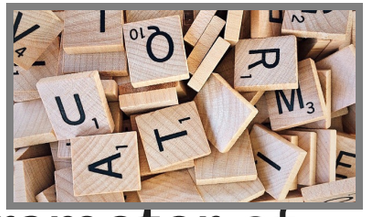
Separate Training, Validation and Test Data

Often a model is "tuned" manually or automatically on a validation set (hyperparameter optimization)

In this case, we can overfit on the validation set, separate test set is needed for final evaluation

```
train_xs, train_ys, valid_xs, valid_ys, test_xs, test_ys =
            split(all_xs, all_ys)

best_model = null
best_model_accuracy = 0
for (hyperparameters in candidate_hyperparameters)
  candidate_model = learn(train_xs, train_ys, hyperparameter)
  model_accuracy = accuracy(model, valid_xs, valid_ys)
  if (model_accuracy > best_model_accuracy)
    best_model = candidate_model
    best_model_accuracy = model_accuracy

accuracy_test = accuracy(model, test_xs, test_ys)
```

On Terminology



- The decisions in a model (weights, coefficients) are called **model parameter** of the model, their values are usually learned from the data
  - To a software engineer, these are **constants** in the learned function
- The inputs to the learning algorithm that are not the data are called **model hyperparameters**
  - To a software engineer, these are **parameters** to the learning algorithm, similar to compiler options

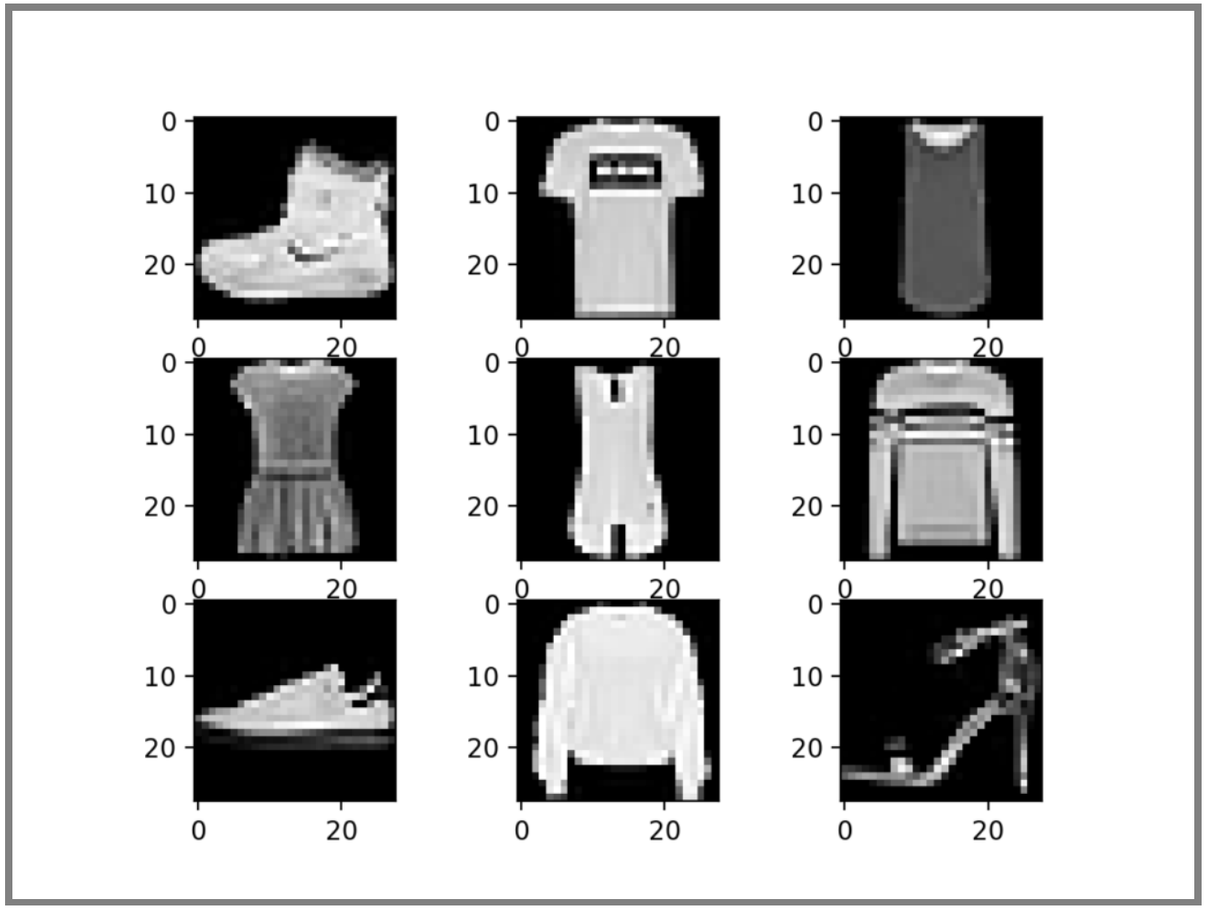
```
// max_depth and min_support are hyperparameters
def learn_decision_tree(data, max_depth, min_support): Model =
  ...
// A, B, C are model parameters of model f
def f(outlook, temperature, humidity, windy) =
   if A==outlook
      return B*temperature + C*windy > 10
```

Common Pitfalls of Evaluating Model Quality

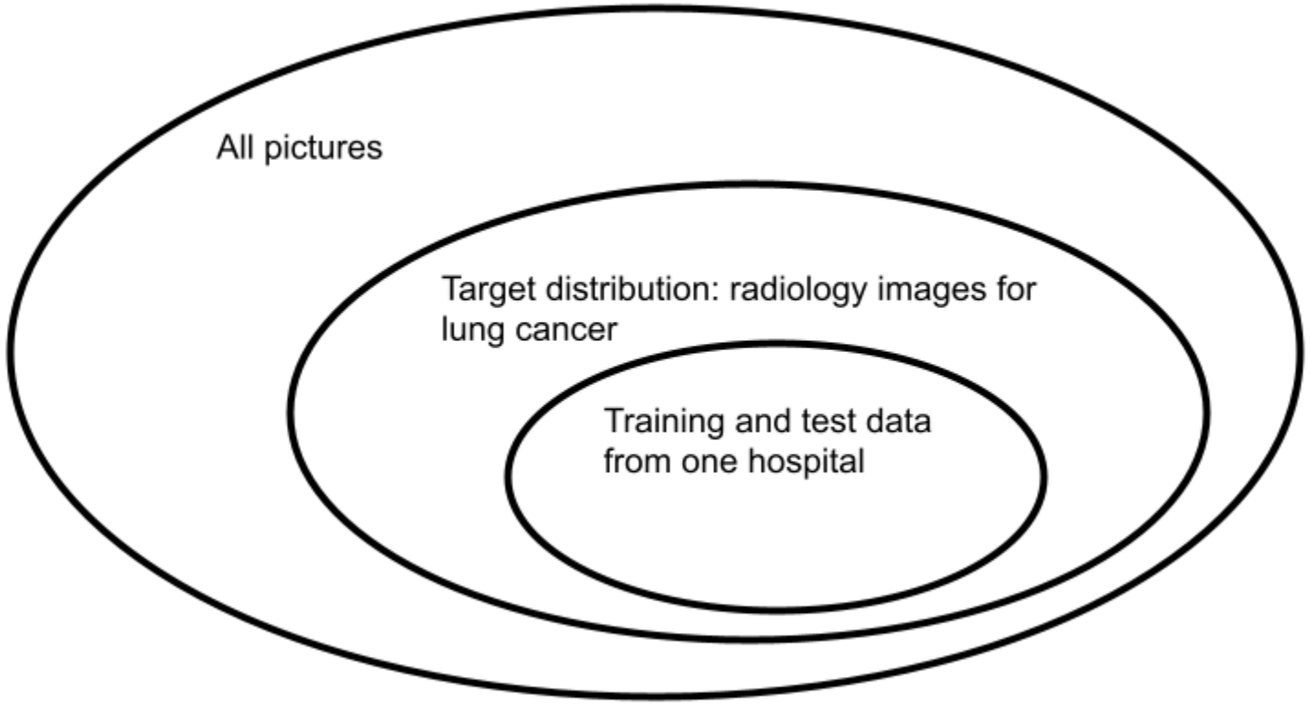
Common Pitfalls of Evaluating Model Quality?

Test Data not Representative

Often neither training nor test data representative of production data



Test Data not Representative



Shortcut Learning

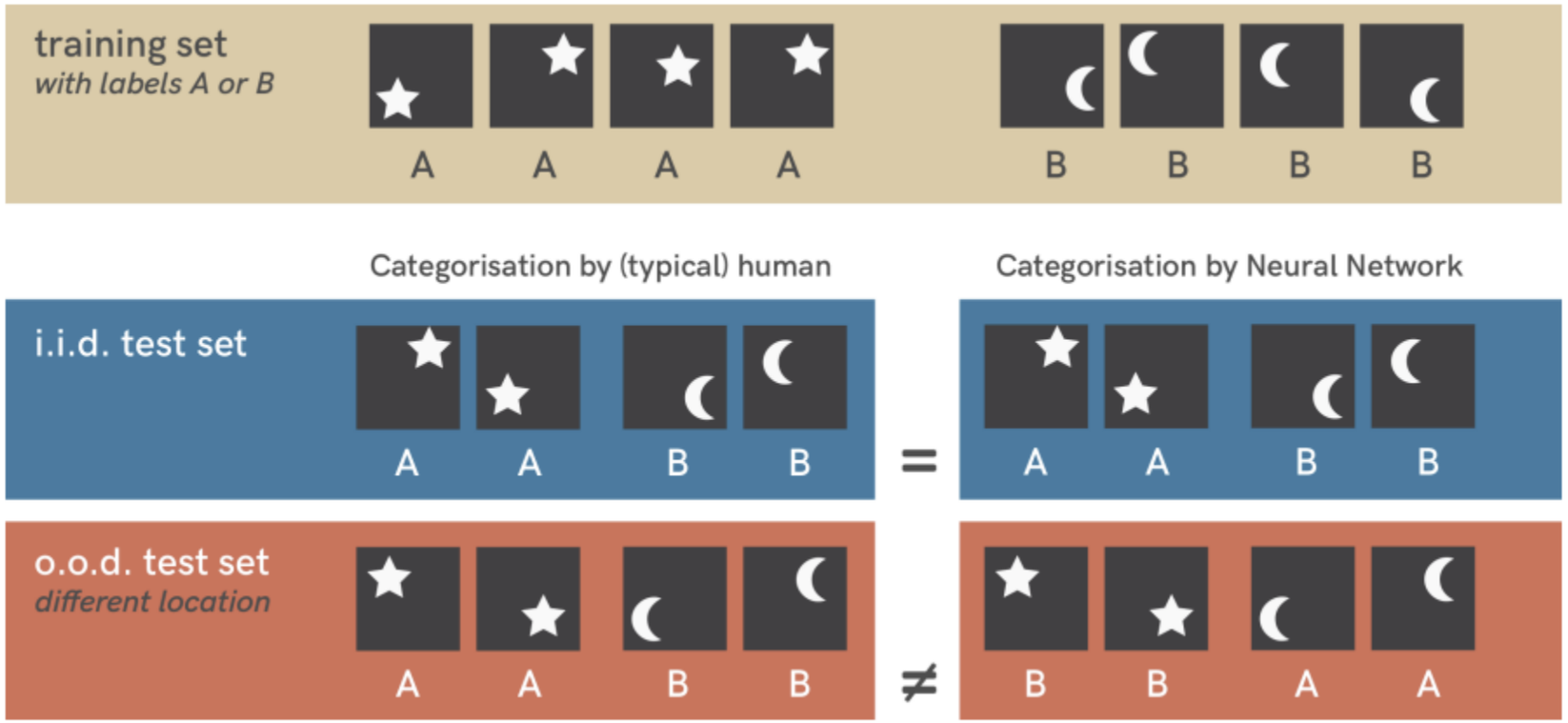


Figure from: Geirhos, Robert, et al. "Shortcut learning in deep neural networks." Nature Machine Intelligence 2, no. 11 (2020): 665-673.

Speaker notes

(From figure caption) Toy example of shortcut learning in neural networks. When trained on a simple dataset of stars and moons (top row), a standard neural network (three layers, fully connected) can easily categorise novel similar exemplars (mathematically termed i.i.d. test set, defined later in Section 3). However, testing it on a slightly different dataset (o.o.d. test set, bottom row) reveals a shortcut strategy: The network has learned to associate object location with a category. During training, stars were always shown in the top right or bottom left of an image; moons in the top left or bottom right. This pattern is still present in samples from the i.i.d. test set (middle row) but not in o.o.d. test images (bottom row), exposing the shortcut.

Shortcut Learning

NeuralTalk2: A flock of birds flying in the air

Microsoft Azure: A group of giraffe standing next to a tree

Image: Fred Dunn, https://www.flickr.com/photos/gratapictures - CC-BY-NC

Other Examples of Shortcut Learning?

Generalization Beyond Training Data:

Is this even fair to ask?

Representative Test Data in Practice?

Target distribution may not be known in early stages of the project

Production data is good test data

Target distribution may shift over time

Monitoring and continuous data collection important! More later

Label Leakage

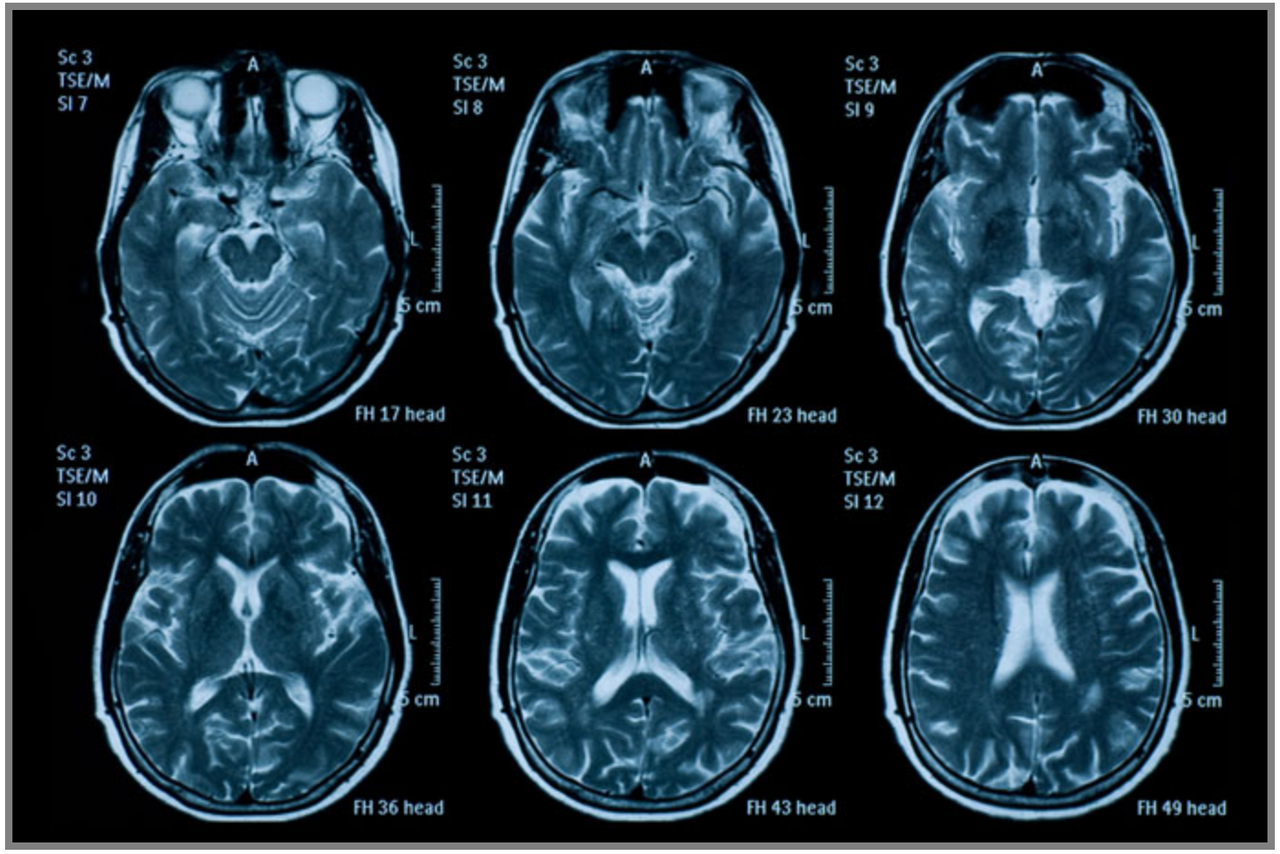
Speaker notes

Widely shared story, authenticity not clear: AI research team tried to train image recognition to identify tanks hidden in forests, trained on images of tanks in forests and images of same or similar forests without tanks. The model could clearly separate the learned pictures, but would perform poorly on other pictures.

Turns out the pictures with tanks were taken on a sunny day whereas the other pictures were taken on a cloudy day. The model picked up on the brightness of the picture rather than the presence of a tank, which worked great for the training set, but did not generalize.

Pictures: https://pixabay.com/photos/lost-places-panzer-wreck-metal-3907364/, https://pixabay.com/photos/forest-dark-woods-trail-path-1031022/

Label Leakage



Speaker notes

The image includes metadata. Models have been found to rely heavily on that metadata, for example what kind of device was used to take the scan (mobile vs stationary) or where it was taken (rural vs big hospital).

Label Leakage

Label or close correlates included in inputs

Examples:

- Input "interview conducted" in turnover prediction encodes human judgement
- Input "has bank account" associates with predicting whether somebody will open one

Is this a problem or a good thing?

Be cautious of "too good to be true" results

Data Leakage: Learning from Test Data

Test data *leaks* into training data

- surprisingly common in practice
- by accident, incorrect split -- or intentional using all data for training
- overlap between multiple datasets used
- data preprocessing on entire dataset
- tuning on validation data (e.g., crossvalidation) without separate testing data

Results in overfitting and misleading accuracy measures

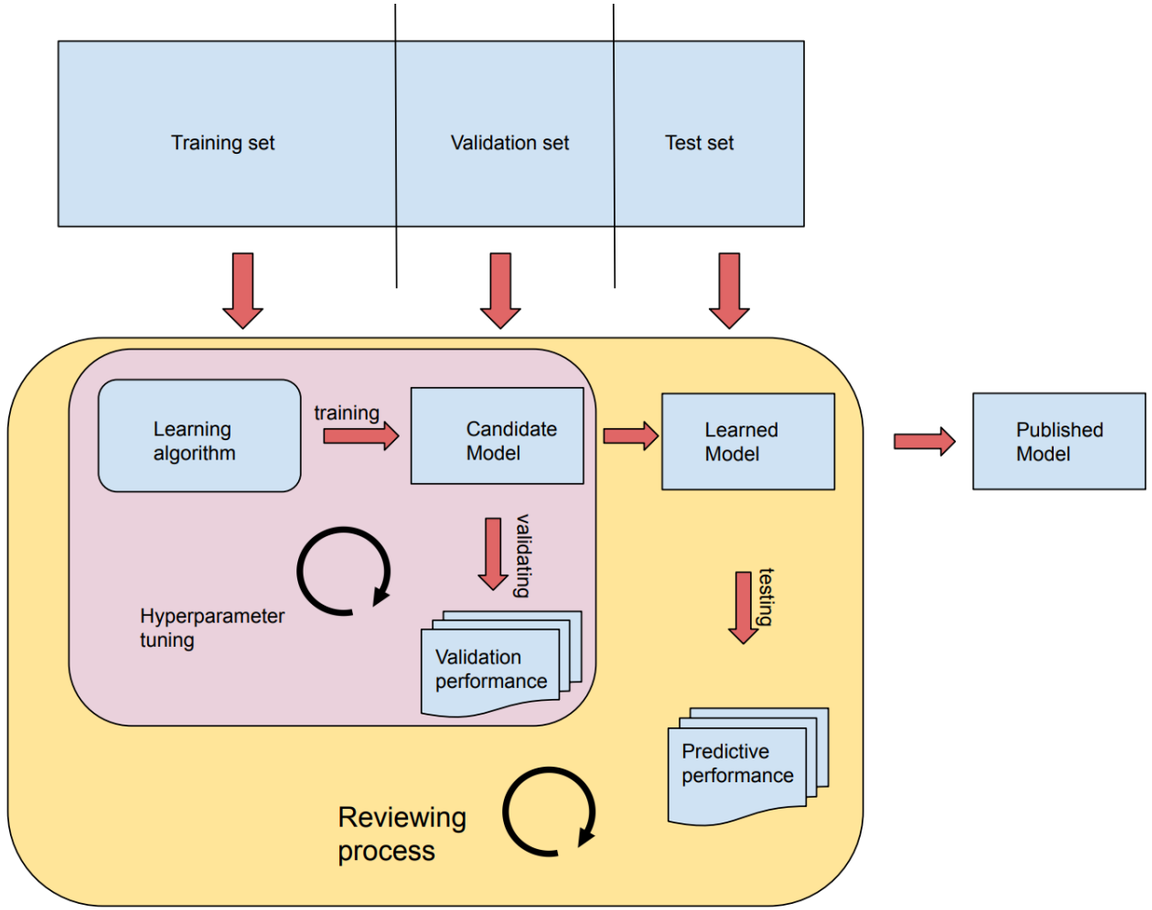
Data Leakage during Data Preprocessing

```
wordsVectorizer = CountVectorizer().fit(text)
wordsVector = wordsVectorizer.transform(text)
invTransformer = TfidfTransformer().fit(wordsVector)
invFreqOfWords = invTransformer.transform(wordsVector)
X = pd.DataFrame(invFreqOfWords.toarray())

train, test, spamLabelTrain, spamLabelTest =
                    train_test_split(X, y, test_size = 0.5)
predictAndReport(train = train, test = test)
```

Data Leakage: Overfitting on Random Data

```
import numpy as np
# generate random data
n_samples, n_features, n_classes = 200, 10000, 2
rng = np.random.RandomState(42)
X = rng.standard_normal((n_samples, n_features))
y = rng.choice(n_classes, n_samples)

# leak test data through feature selection
X_selected = SelectKBest(k=25).fit_transform(X, y)

X_train, X_test, y_train, y_test =
```

Overfitting on Benchmarks

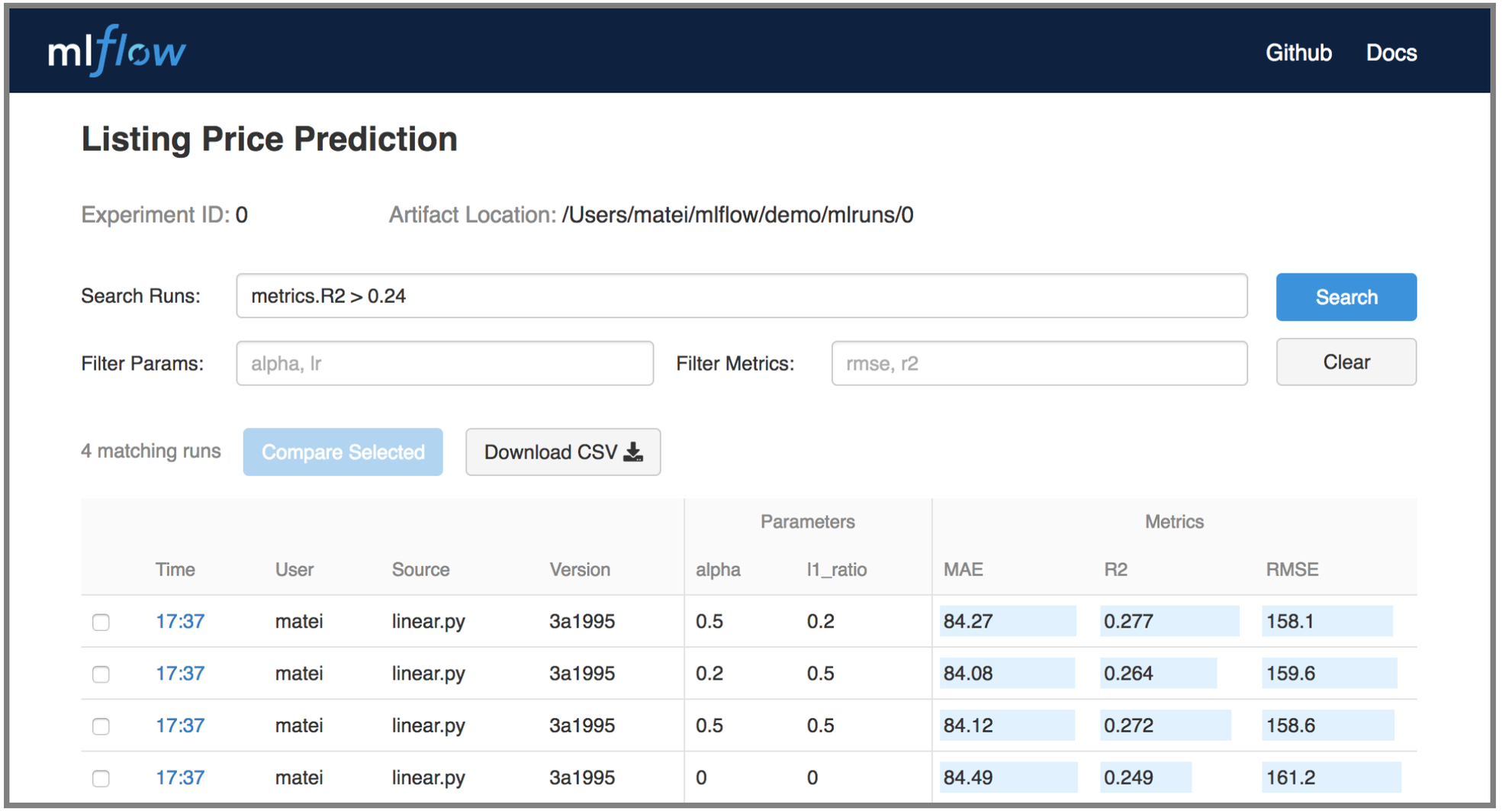


(Figure by Andrea Passerini)

Speaker notes

If many researchers publish best results on the same benchmark, collectively they perform "hyperparameter optimization" on the test set

Overfitting in Continuous Experimentation



Overfitting in Continuous Experimentation

- Test data should be used exactly once -- danger of overfitting with reuse
- Use of test sets to compare (hyperparameter-tuned) models in dashboards ➤ danger of overfitting
- Need fresh test data regularly
- Statistical techniques to approximate the needed amount of test data and the needed rotation

Recommended reading: Renggli, Cedric, Bojan Karlaš, Bolin Ding, Feng Liu, Kevin Schawinski, Wentao Wu, and Ce Zhang. "Continuous integration of machine learning models with ease.ml/ci:

Using Misleading Quality Measures

- using accuracy, when false positives are more harmful than false negatives
- comparing area under the curve, rather than relevant thresholds
- averaging over all populations, ignoring different results for subpopulations or different risks for certain predictions
- accuracy results on old static test data, when production data has shifted
- results on tiny validation sets
- reporting results without baseline
- ...

Independence of Data: Temporal

Attempt to predict the stock price development for different companies based on twitter posts

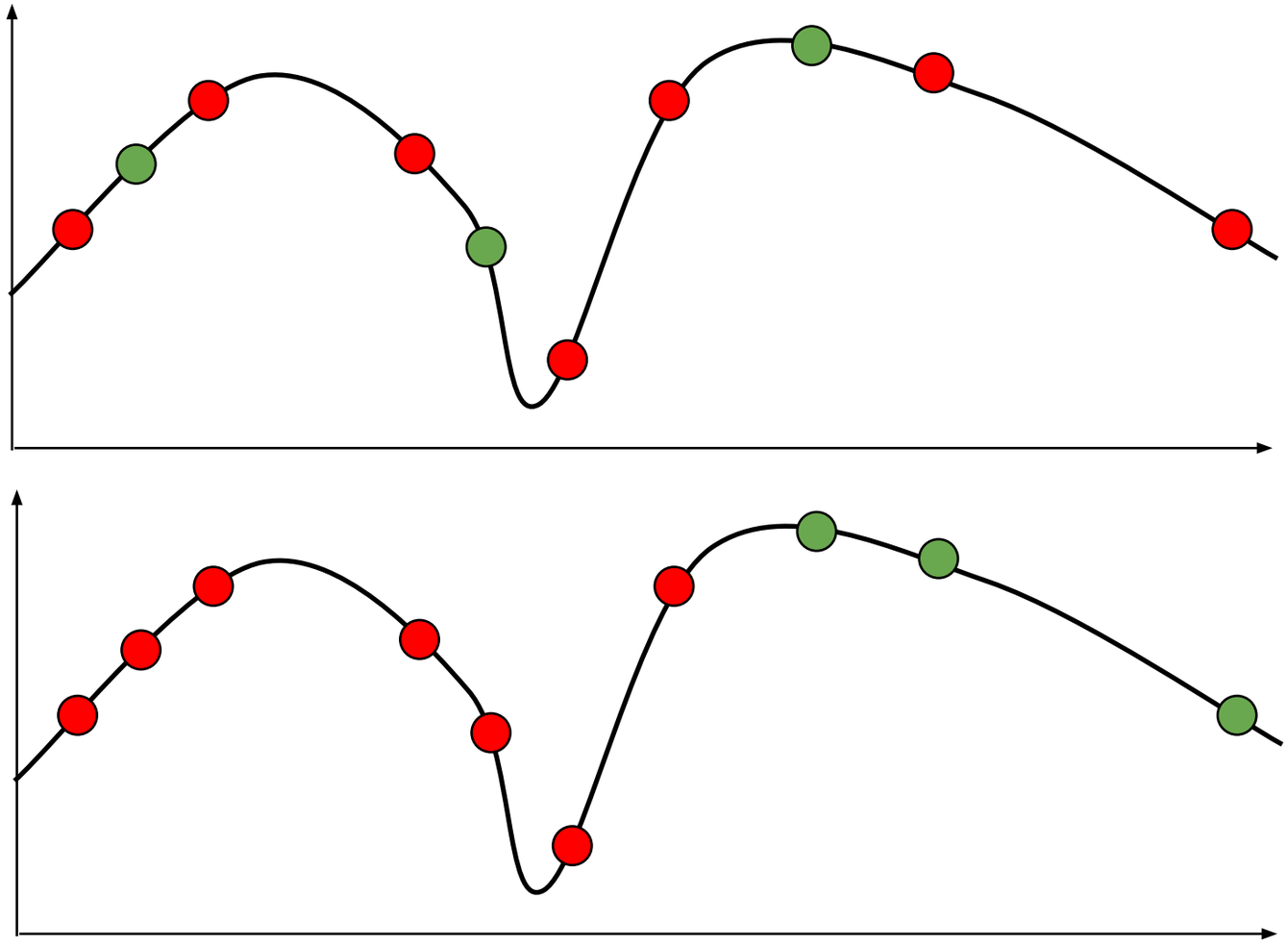
Data: stock prices of 1000 companies over 4 years and twitter mentions of those companies

Problems of random train--validation split?

Speaker notes

The model will be evaluated on past stock prices knowing the future prices of the companies in the training set. Even if we split by companies, we could observe general future trends in the economy during training

Independence of Data: Temporal



Speaker notes

The curve is the real trend, red points are training data, green points are validation data. If validation data is randomly selected, it is much easier to predict, because the trends around it are known.

Independence of Data: Related Datapoints

Example: Kaggle competition on detecting distracted drivers

Driver Picture 1

Relation of datapoints may not be in the data (e.g., driver)

https://www.fast.ai/2017/11/13/validation-sets/

Speaker notes

Many potential subtle and less subtle problems:

- Sales from same user
- Pictures taken on same day

Data Dependence in Cancer Case Study?

Preliminary Summary: Common Pitfalls

- **Always question the i.i.d. assumption**
- Test data not representative
- Dependence between training and test data
- Misleading accuracy metrics
- Evaluating on training or validation data
- Label leakage
- Overfitting on test data through repeated evaluations

How to avoid? Ensure as part of process?

Speaker notes

i.i.d. = independent and identically distributed

Part 2:

What is Correctness Anyway?

specifications, bugs, fit

SE World: Evaluating a Component's Functional Correctness

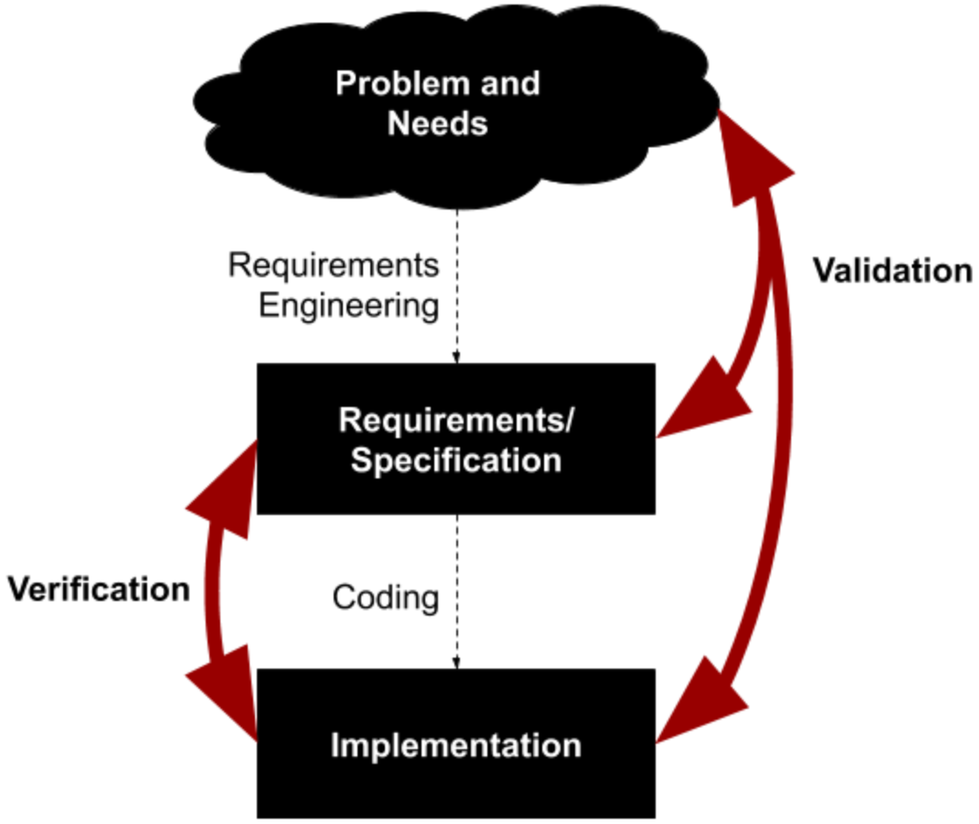
Given a specification, do outputs match inputs?

```
/**
 * compute deductions based on provided adjusted
 * gross income and expenses in customer data.
 *
 * see tax code 26 U.S. Code A.1.B, PART VI
 */
float computeDeductions(float agi, Expenses expenses);
```

Each mismatch is considered a bug, should to be fixed.†

(†=not every bug is economical to fix, may accept some known bugs)

Validation vs Verification



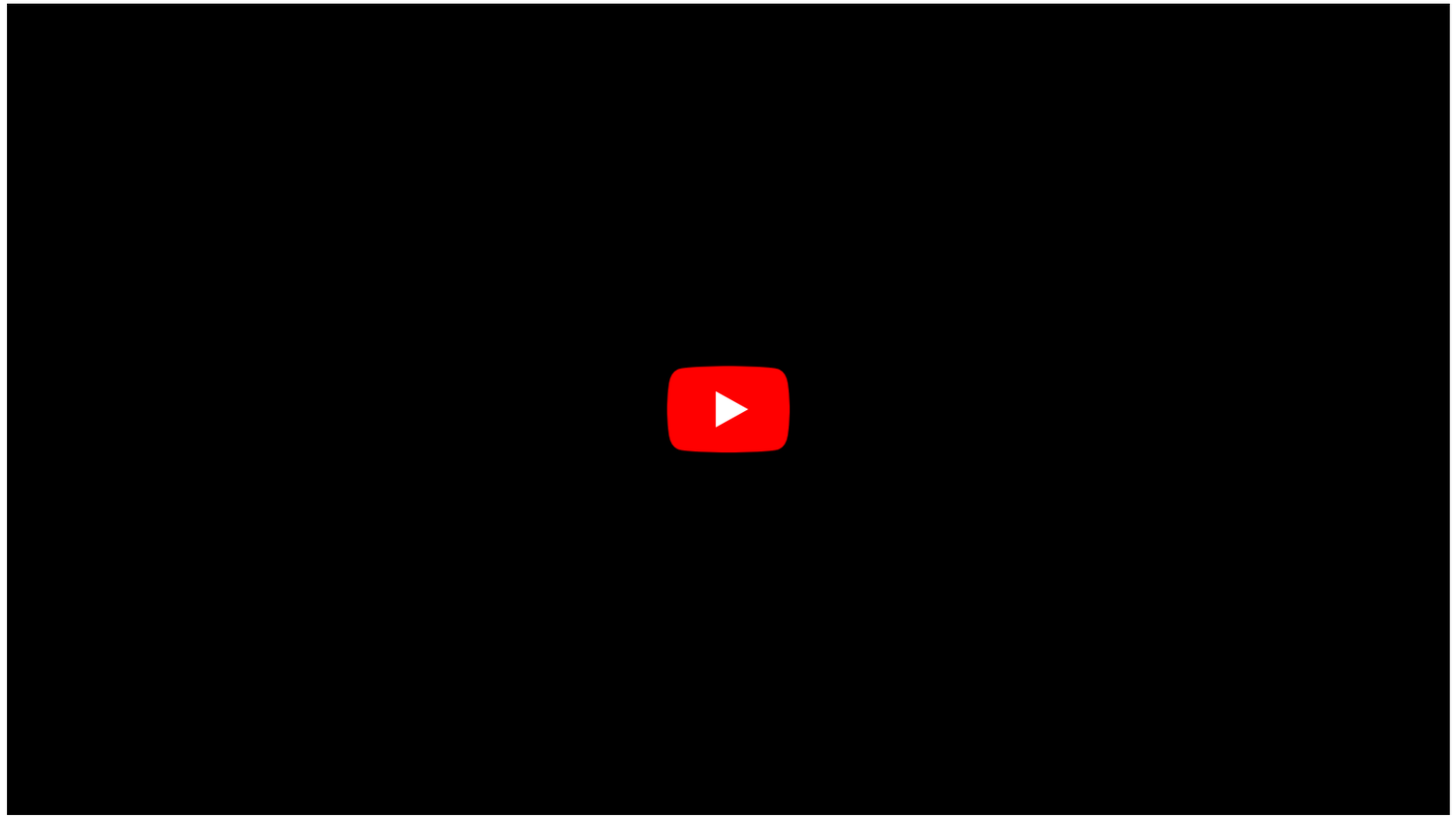
Validation Problem: Correct but Useless?

- Correctly implemented to specification, but specifications are wrong
- Building the wrong system, not what user needs
- Ignoring assumptions about how the system is used

Example: Compute deductions with last year's tax code

Other examples?

Wrong Specifications: Ariane 5



Software was working as specified, within the specified parameters.
Inputs exceeded specified parameters.

Strict Correctness Assumption

- Specification determines which outputs are correct/wrong
- Not "pretty good", "95% accurate", or "correct for 98% of all users"
- A single wrong result indicates a bug in the system

```
/**
 * compute deductions based on provided adjusted
 * gross income and expenses in customer data.
 *
 * see tax code 26 U.S. Code A.1.B, PART VI
 */
float computeDeductions(float agi, Expenses expenses);
```

Speaker notes

A single wrong tax prediction would be a bug. No tolerance of occasional wrong predictions, approximations, nondeterminism.

Ideally formal specifications

Formal verification possible, proving that implementation matches specification.

```
/*@ public normal_behavior
  @ ensures (\forall int j; j >= 0 && j < a.length;
  @                          \result = a[j]);
  @*/
public static /*@ pure @*/ int max(int[] a);
```

In practice, typically informal, textual and "incomplete" specifications, but still enabling analyzing inputs-output correspondence

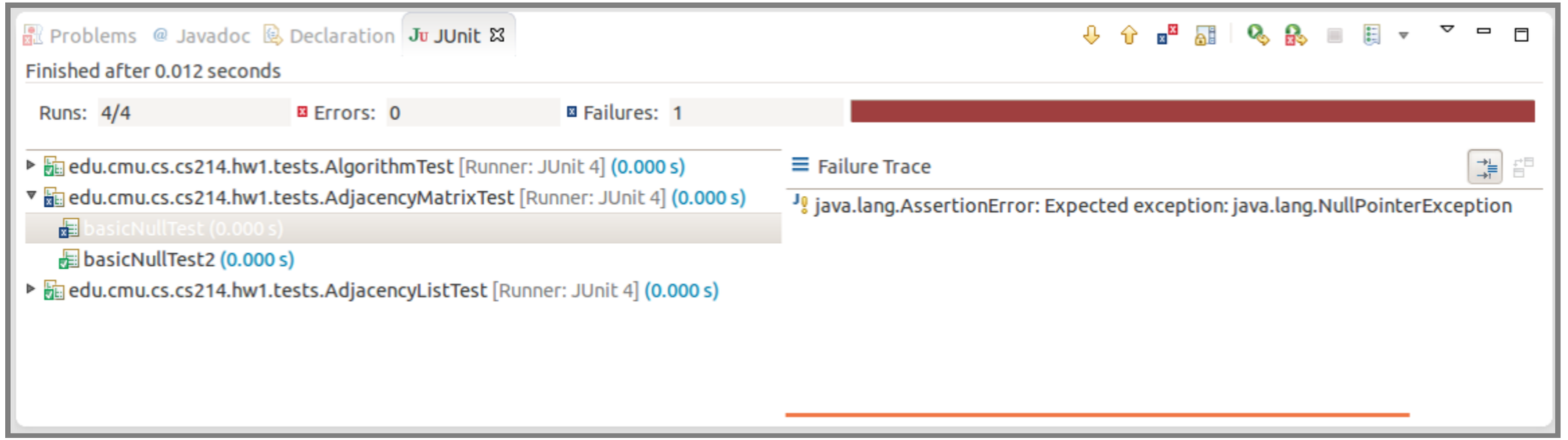
Common practice: Testing

- Verification technique comparing program behavior to specification
- Provide select inputs, expect correct outputs (according to specification)
- Failing test case reveals bug
- No guarantee to find all bugs

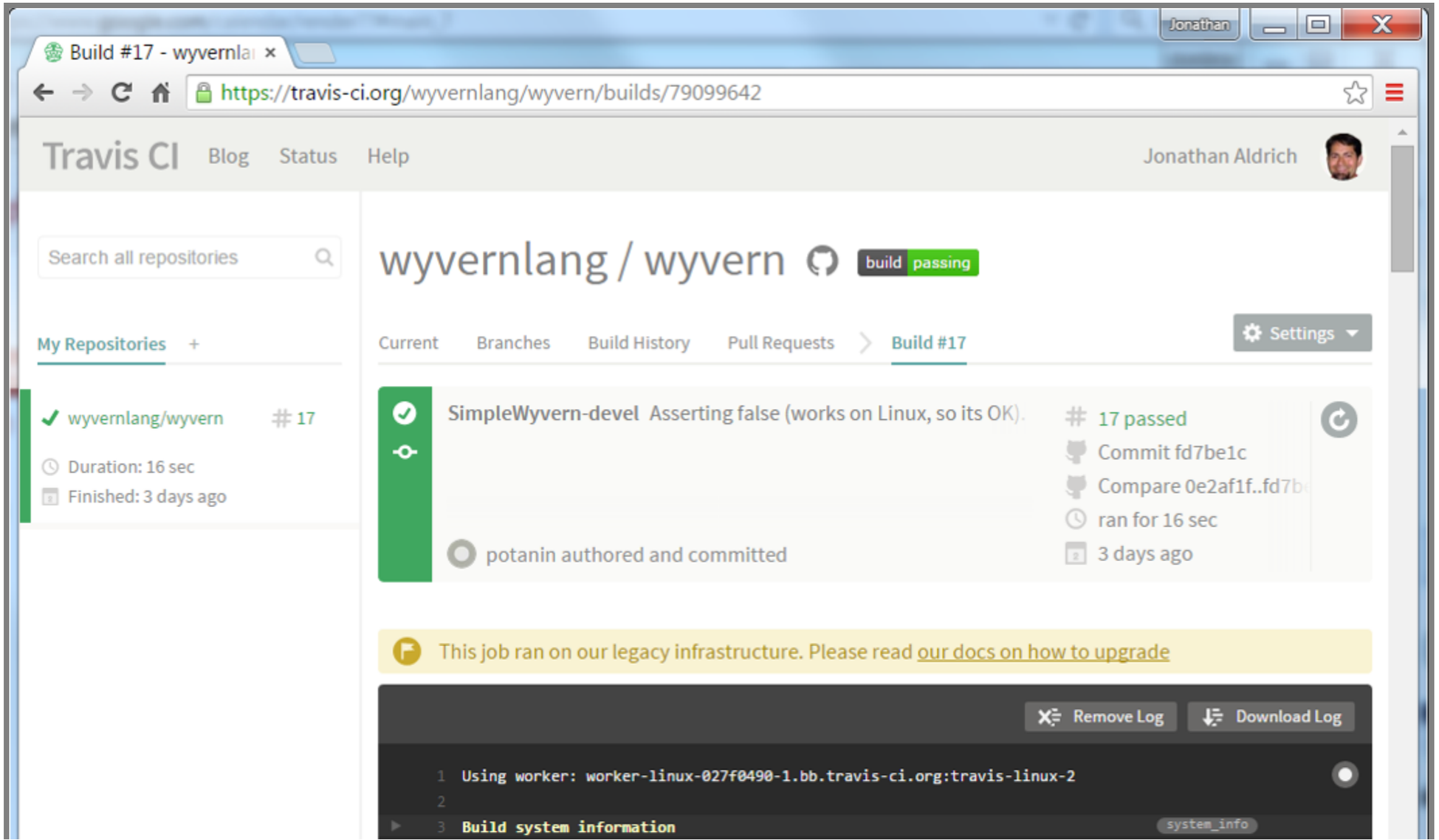
```
// returns the sum of two arguments
int add(int a, int b) { ... }

@Test
void testAddition_2_2() {
  assertEquals(4, add(2, 2));
}
@Test
void testAddition_1_2() {
  assertEquals(3, add(1, 2));
}
```

Test automation

```
@Test
void testAddition_2_2() {
  assertEquals(4, add(2, 2));
}
```



Continuous Integration



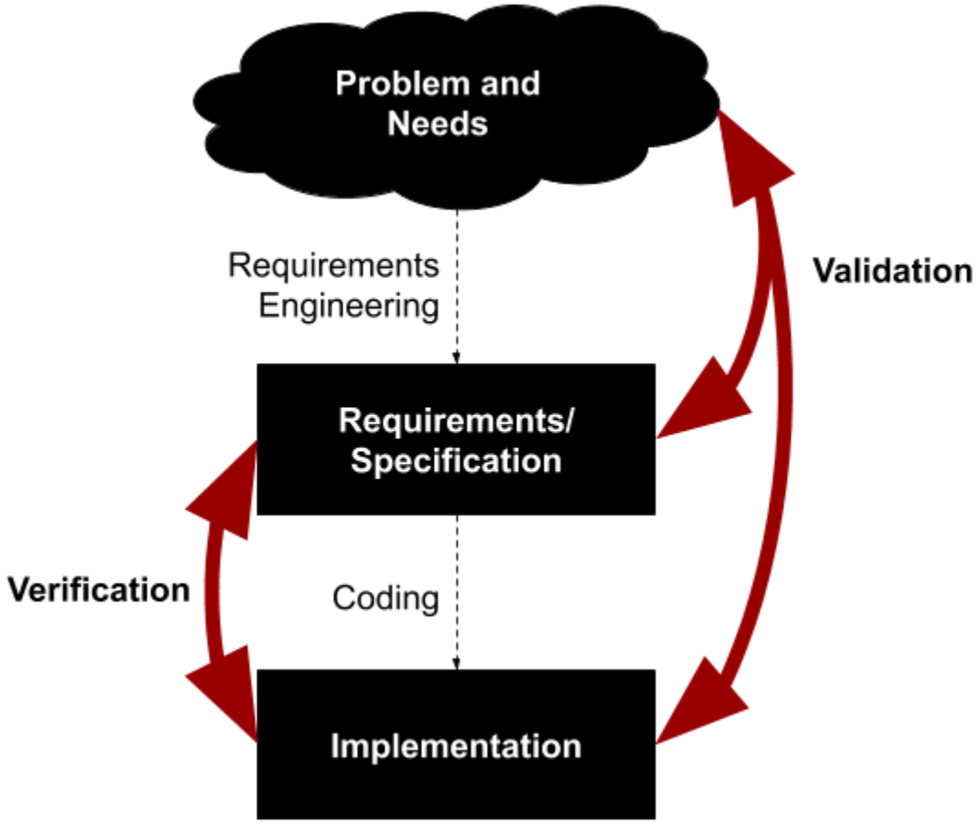
Software Testing

> *"Testing shows the presence, not the absence of bugs" -- Edsger W. Dijkstra 1969*

Software testing can be applied to many qualities:

- Functional errors
- Performance errors
- Buffer overflows
- Usability errors
- Robustness errors
- Hardware errors
- API usage errors

Validation vs Verification



Requirements Validation

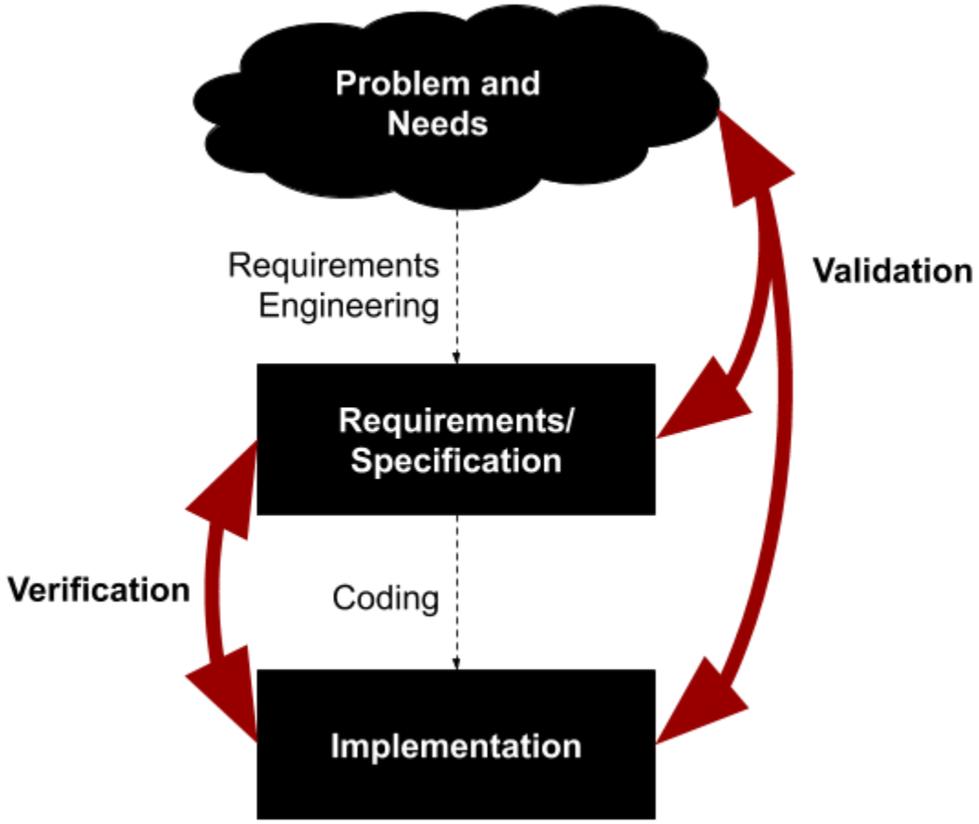
Talk to stakeholders

Build prototype, show to potential users

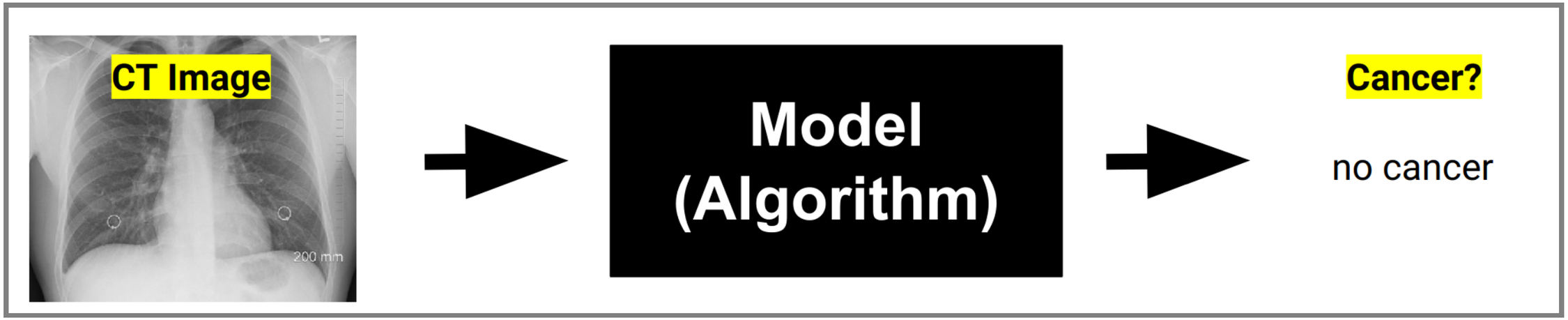
Involve customer in design discussions (agile)

Ask experts whether requirements cover important concerns, check legal compliance

Validation vs Verification

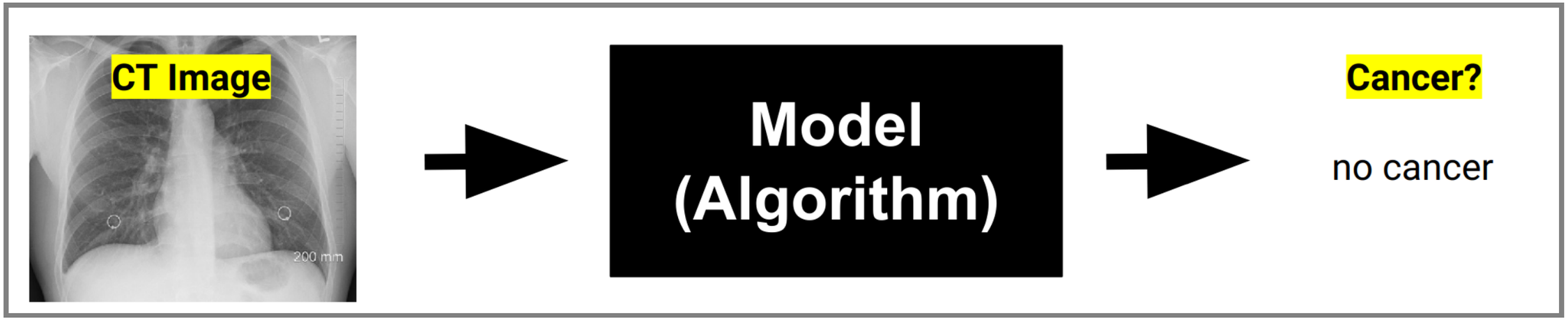


How to evaluate prediction tasks?




```
/**
   ????
*/
boolean hasCancer(Image scan);
```

No specification!



Use ML precisely because no specifications (too complex, rules unknown)

- No specification that could tell us for any input whether the output is correct
- Intuitions, ideas, goals, examples, "implicit specifications", but nothing we can write down as rules!
- *We are usually okay with some wrong predictions*

Testing a Machine Learning Model?

```
// detects cancer in an image
boolean hasCancer(Image scan);

@Test
void testPatient1() {
  assertEquals(loadImage("patient1.jpg"), false);
}
@Test
void testPatient2() {
  assertEquals(loadImage("patient2.jpg"), false);
}
```

Weak Correctness Assumptions

- Often no reliable ground truth (e.g. human judgment and disagreement)
- Examples, but no rules
- Accepting that mistakes will happen, hopefully not to frequently; "95% accuracy" may be pretty good
- More confident for data similar to training data

All Models Are Wrong

> *All models are approximations. Assumptions, whether implied or clearly stated, are never exactly true.* ***All models are wrong, but some models are useful****. So the question you need to ask is not "Is the model true?" (it never is) but "Is the model good enough for this particular application?" -- George Box*

Non-ML Example: Newton's Laws of Motion

> *2nd law: "the rate of change of momentum of a body over time is directly proportional to the force applied, and occurs in the same direction as the applied force"* $\mathbf{F} = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}$

"Newton's laws were verified by experiment and observation for over 200 years, and they are excellent approximations at the scales and speeds of everyday life."

Do not generalize for very small scales, very high speeds, or in very strong gravitational fields. Do not explain semiconductor, GPS errors, superconductivity, ... Those require general relativity and quantum field theory.

All Models Are Wrong

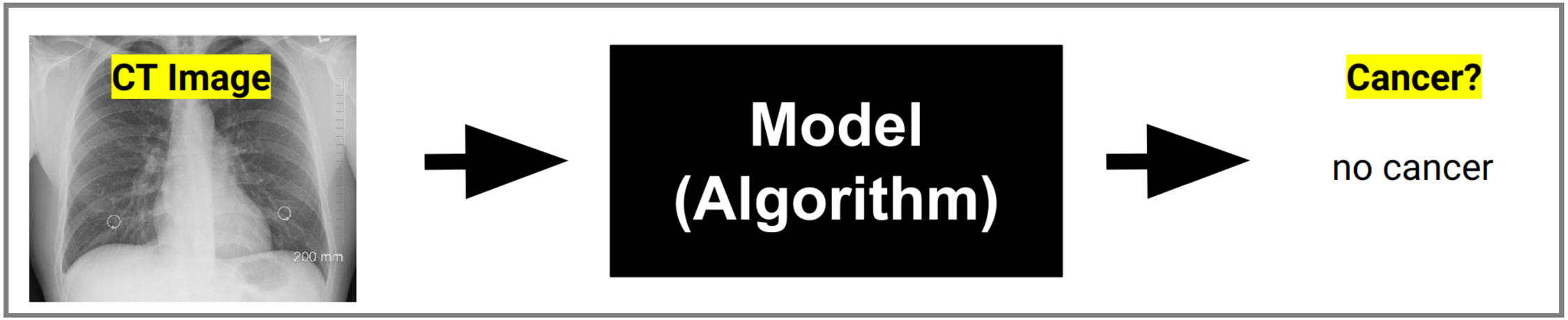
"Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena." -- George Box, 1976

"Since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad." -- George Box, 1976

Does Knowledge Empower Us?

> *Knowledge is power: The real test of "knowledge" is not whether it is true, but whether it empowers us. Scientists usually assume that no theory is 100 per cent correct. Consequently, truth is a poor test for knowledge. The real test is utility. A theory that enables us to do new things constitutes knowledge. -- Yuval Harari in Sapiens about Francis Bacon's "New Instrument" from 1620*

Find better models?



We are looking for models that better **fit** the problem

No specification of "correctness"

Some wrong predictions accepted, not too many though

Typical accuracy evaluation

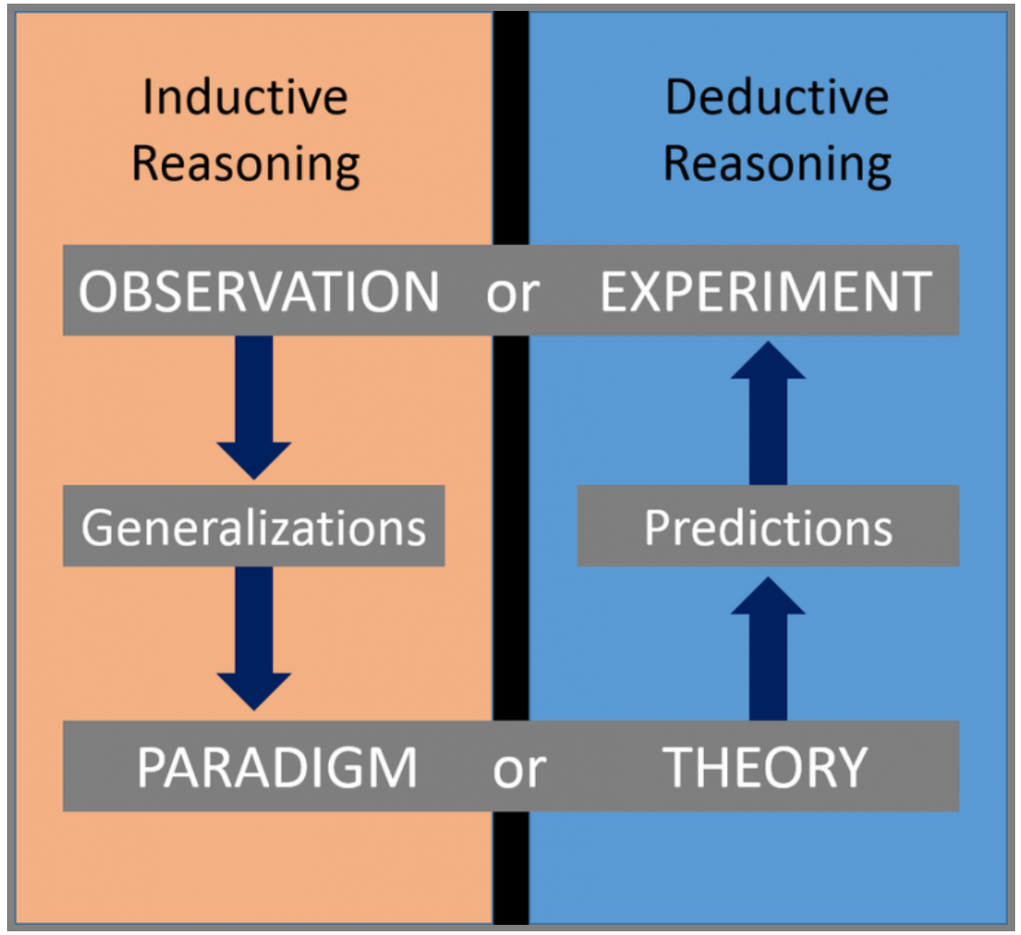
Given example data, evaluate how well the model *fits* that data

```
def accuracy(model, xs, ys):
  count = length(xs)
  countCorrect = 0
  for i in 1..count:
    predicted = model(xs[i])
    if predicted == ys[i]:
      countCorrect += 1
  return countCorrect / count
```

Only sample inputs, like in software testing

Unlike traditional software *do not expect "correctness"*

Deductive vs Inductive Reasoning



(Daniel Miessler, CC SA 2.0)

Inductive Reasoning

- Constructing axioms from observations
- Strong evidence suggests a rule
- From particular to the general
- *sciency reasoning, eg. finding laws of nature*

- Most modern machine learning systems, statistical learning

Deductive Reason.

- Combining logical statements following agreed upon rules to form new statements
- Proving theorems from axioms
- From general to the particular
- *mathy reasoning, eg. proof that π is irrational*

- Formal methods, classic rule-based AI systems, expert systems

Machine Learning Models Fit, or Not

- A model is learned from given data in given procedure
  - The learning process is typically not a correctness concern
  - The model itself is generated, typically no implementation issues
- Is the data representative? Sufficient? High quality?
- Does the model "learn" meaningful concepts?

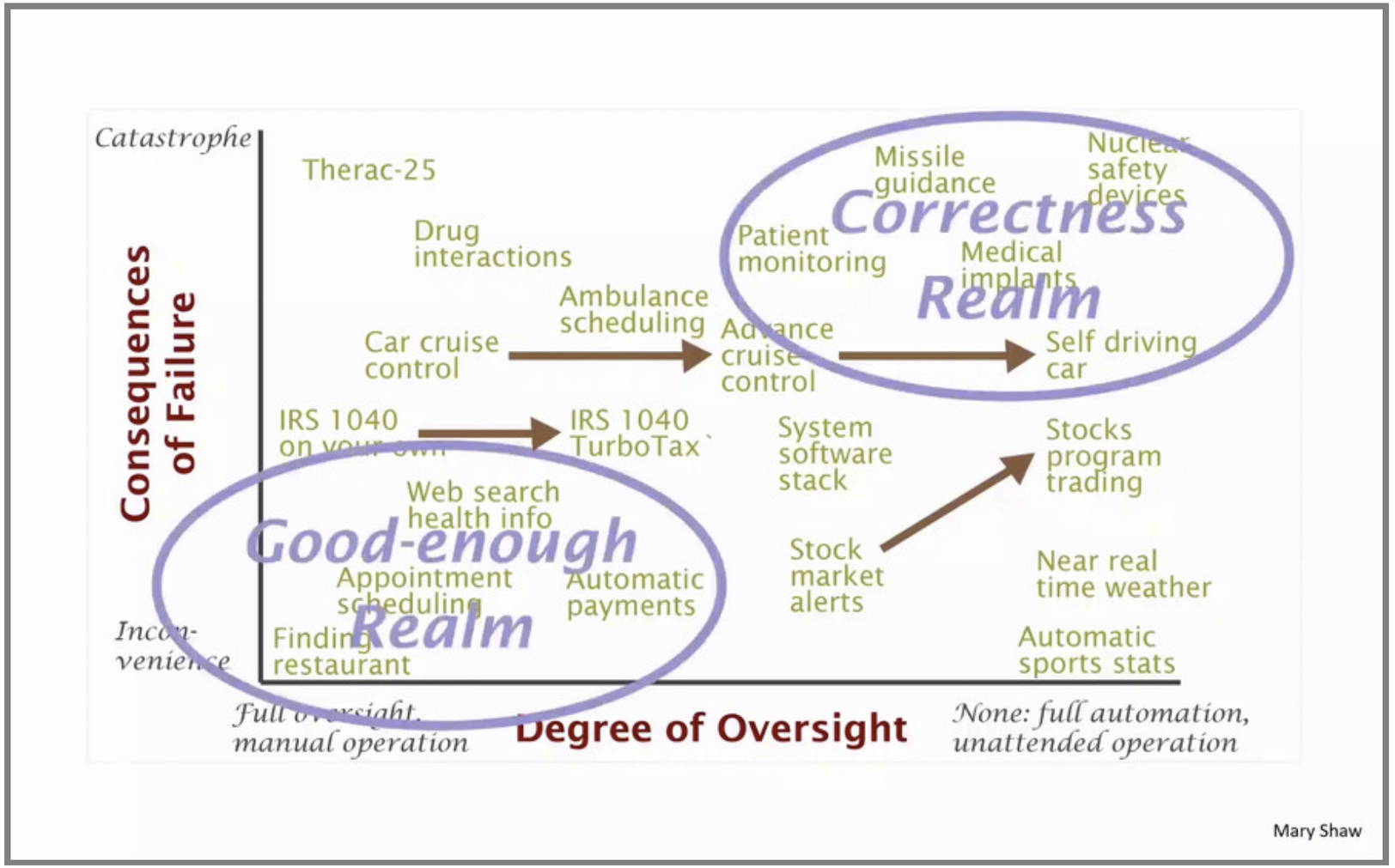
- **Is the model useful for a problem?** Does it *fit*?
- Do model predictions *usually* fit the users' expectations?
- Is the model *consistent* with other requirements? (e.g., fairness, robustness)

My pet theory: Machine Learning is Requirements Engineering

Software Engineering Caveat

- Also software engineers rarely assure "correctness"
- Testing finds bugs, does not assure their absence
- Formal verification possible, but expensive and rare
- Real challenges involve interactions with environment, which are hard to specify

- "Good enough" very common for software quality
- Evaluating "fit for intended purpose" instead of correctness too



Mary Shaw. Myths and Mythconceptions: What does it mean to be a programming language, anyhow? HOPL IV: History of Programming Languages, 2021.

On Terminology

- Avoid term **model bug**, no agreement, no standardization
- **Performance** or **accuracy** or **fit** are better fitting terms than *correct* for model quality
- Careful with the term **testing** for measuring **prediction accuracy**, be aware of "correctness" connotations
- *Verification/validation* analogy may help frame thinking, but will likely be confusing without longer explanation

Summary

Model prediction accuracy only one part of system quality

Select suitable measure for prediction accuracy, depending on problem

Use baselines for interpreting prediction accuracy

Avoid common pitfalls in evaluating model accuracy

"Software bugs" vs "model fit" in the absence of specifications -- all models are wrong

Further readings

- Kaestner, Christian. "Machine Learning is Requirements Engineering — On the Role of Bugs, Verification, and Validation in Machine Learning." Medium Blog Post. 2020.