

Principles of AI Engineering

Chapter 3: Model Quality

Prof. Dr. Steffen Herbold

Credit:

Based on contents from Christian Kästner (<https://github.com/ckaestne/seai>)

Contents

- What is model quality?
- Model quality from a data science perspective
- Evaluating generalization
- Pitfalls of evaluations
- Correctness from a software engineering perspective
- Correctness of machine learning from a software engineering perspective
- Learning from software testing
- The oracle problem
- Metamorphic testing
- More software testing concepts

What is model quality?

Case study: Cancer prognosis



Radiologist analyze MRIs and other medical data to understand cancer



Geoffrey Hinton

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

<https://www.youtube.com/watch?v=2HMPRXstSvQ&t=29s>

This statement was made in 2016!

Model is part of a system in an environment

Tryton - Administrator - GNU SOLIDARIO HOSPITAL [Euro]

File User Options Favorites Help

screen

- Addresses
- Categories
- Product
- Financial
- Currency
- Inventory & Stock
- Purchase
- Calendar
- Health
- Patients**
- Institutions
- Appointments
- Prescriptions
- Demographics
- Laboratory
- Imaging
- Hospitalizations
- Surgeries
- Pediatrics
- Archives
- Nursing
- Health Services
- Reporting
- Configuration

Patients

Obstetric Hist ...

1 / 8

New Save Switch Reload Previous Next Attachment(0) Action Relate Report E-Mail Print

Main Info

Betz, Ana Female Age: 29y 3m 20d

Critical Information

Personal history of allergy to penicillin
Insulin-dependent diabetes mellitus

Severe allergic reactions to β -lactams

General Info Socioeconomics Medication Diseases Surgeries Genetics Lifestyle QB/GYN

General Screening

Fertile: ☒ Pregnant: ☐ Menarche age: 12 Menopausal: ☐ Menopause age:

OB summary

Pregnancies: 1 Premature: 0 Abortions: 0 Stillbirths: 0

Menstrual History

Date	LMP	Length	frequency	volume	Regular	Dysmenorrhea	Reviewed	Institution
01/24/2015	01/20/2015	5	eumenorrhea	normal	<input type="checkbox"/>	<input type="checkbox"/>	Cordara, Cameron	GNU SOLIDARIO HOSPITAL

tryton://health.gnusolidario.org:8000/health28rc1/model/gnuhealth.patient/1;views=%5B223%2C+224%5D

https://commons.wikimedia.org/wiki/Category:GNU_Health#/media/File:Gnu_health_2-8_gynecology_general.png

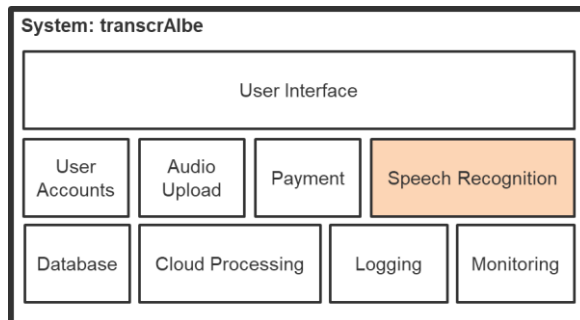
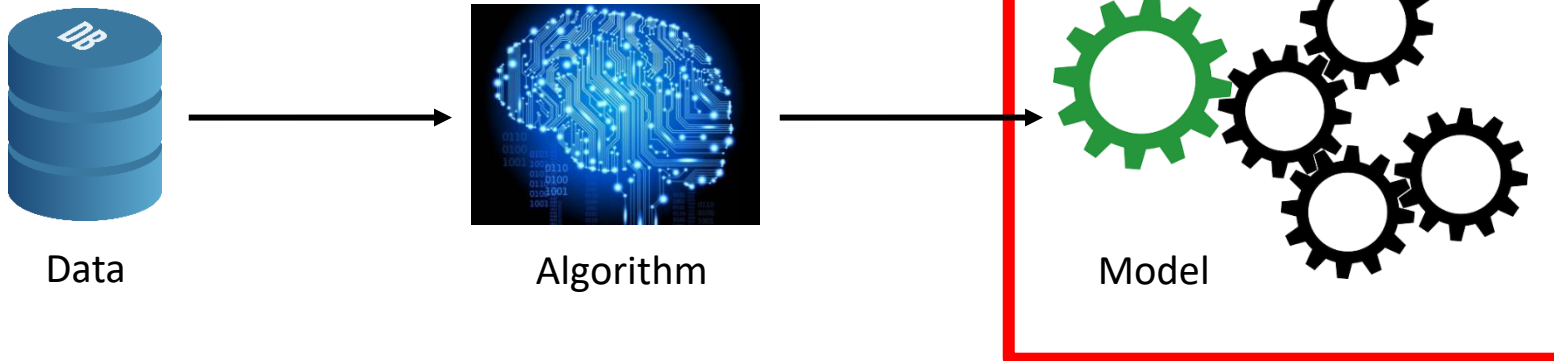
ML algorithm quality vs. model quality vs. data quality vs. system quality

- Many aspects affect the overall quality!
- ML algorithm quality
 - Is the ML algorithm implemented correctly and does it yield good results?
- Model quality
 - Does the trained model yield correct results?
- Data quality
 - Are there issues with the data used for training, validation, and testing?
- System quality
 - Does the overall system fulfill the requirements?

Model quality

Focuses only on the produced model

Checks quality, does not debug source (e.g., algorithm or data)!



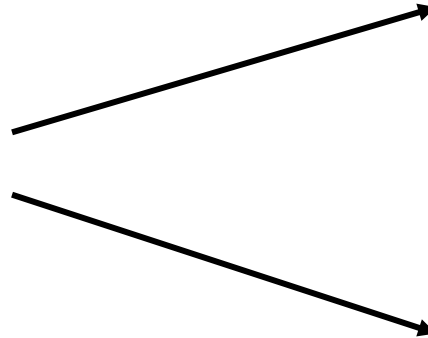
→ Focus is on a small part of the overall system!

Some system-level aspects still important!

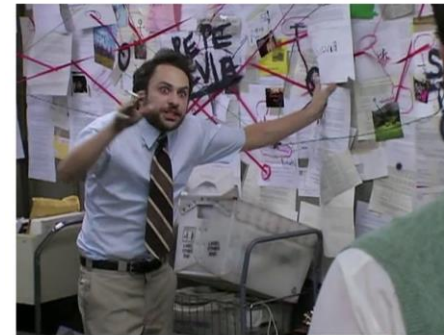
- Usually derived from the use case
- Example:



Radiologist uses model output
→ Human-in-the-Loop



May not trust model



Needs to explain diagnosis
to patients

Many model qualities

- Accuracy: How often are the predictions correct? Which mistake does the model make?
- Model size: How large is the model?
- Inference latency: How long do you need to wait for an inference (prediction)?
- Learning latency: How long do you need to wait for a re-training to finish?
- User interaction model: How can the model interact with users, resp. which interaction modes are supported?
- Ability to incrementally learn: Can the model be easily be updated with new incoming data?
- Explainability: Can the predictions be explained and how complex is the explanation?
- Calibration: Can the model be tuned towards favoring certain aspects?
- Robustness: Is the model robust against adversarial inputs?

Model quality from a data science perspective

Focus on accuracy

The confusion 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

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

$$accuracy = \frac{10 + 24 + 82}{10 + 6 + 2 + 3 + 24 + 10 + 5 + 22 + 82} = 0.707$$

Model quality with respect to system goals

- Suppose you have two models for the same task
- Which model supports the system goals best?
- Which one makes fewer *important* mistakes?
- Which one is easier to operate?
- Which one is better overall?
- Is either one good enough?!



Live exercise: How would you evaluate this for the cancer prognosis example?

99% accuracy = good? 10% accuracy = bad?

- Depends!
- Object detection of other people on the road for a self-driving car?
 - 99% is bad!
- Information retrieval task like a search engine?
 - 10% is good (as long as relevant hits are contained and ranked high)



Is 1% improvement good?

- Depends!
 - From 98% to 99%?
 - From 50% to 60%?
-
- Calculate improvement over baseline:

$$\text{reduction of error} = \frac{\text{accuracy} - \text{accuracy}_{\text{baseline}}}{1 - \text{accuracy}_{\text{baseline}}}$$

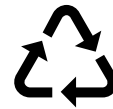
$$\text{reduction of error} = \frac{99\% - 98\%}{1 - 98\%} = 50\%$$

$$\text{reduction of error} = \frac{60\% - 50\%}{1 - 50\%} = 20\%$$



Possible baselines

- Random prediction
- Everything true
- Everything false
- Repeat last observation
- Simple heuristic
 - E.g., threshold on single feature
- Simple model
 - E.g., logistic regression
- State-of-the-art model



Do not forget event likelihood!

- Not every event is likely to happen
- About 1 in 2,000 has cancer

Random model

	Actually Cancer	Actually No Cancer
Model predicts Cancer	3	4998
Model predicts No Cancer	2	4997

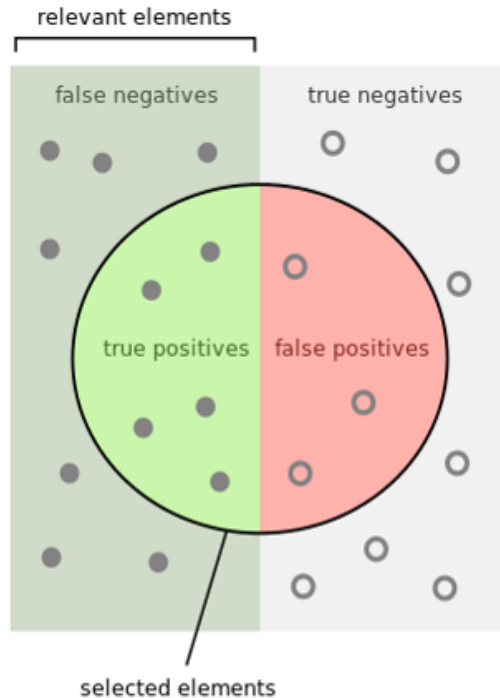
accuracy = 0.5%

Never cancer

	Actually Cancer	Actually No Cancer
Model predicts Cancer	0	0
Model predicts No Cancer	5	9995

accuracy = 99.9%

Different metrics = Different perspectives



Recall measures hit rate of positive predictions

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

Precision measures noise within positive predictions

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

F-measure combines both through the harmonic mean

$$F - \text{measure} = 2 \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$$

Many more!

AUC, TPR (=Recall), TNR, FNR, FPR, MCC, Log loss, ...

Baselines, Recall, and Precision

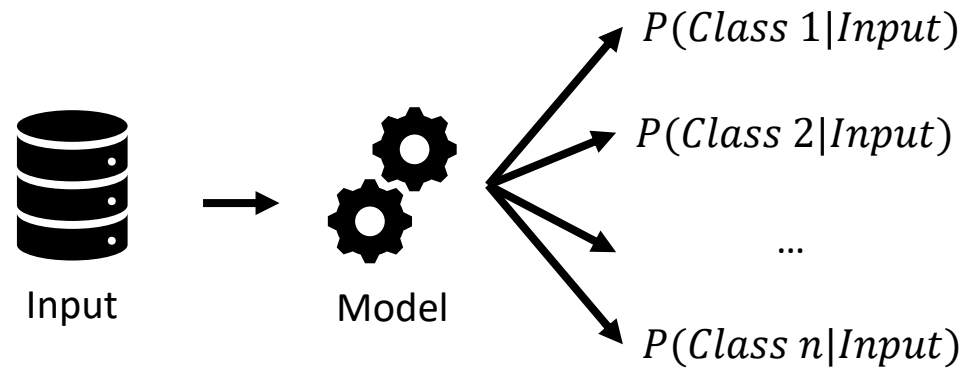
- Predict everything as true
 - Recall = 1
 - Precision = Low value (depends on percentage of false)
- Prediction everything as false
 - Recall = 0
 - Precision = Undefined
- Predict everything randomly
 - Recall = approx. 0.5
 - Precision = approx. 0.5

Live exercise:

Are 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
- Approving loan applications
- What is better? No answer or the wrong answer

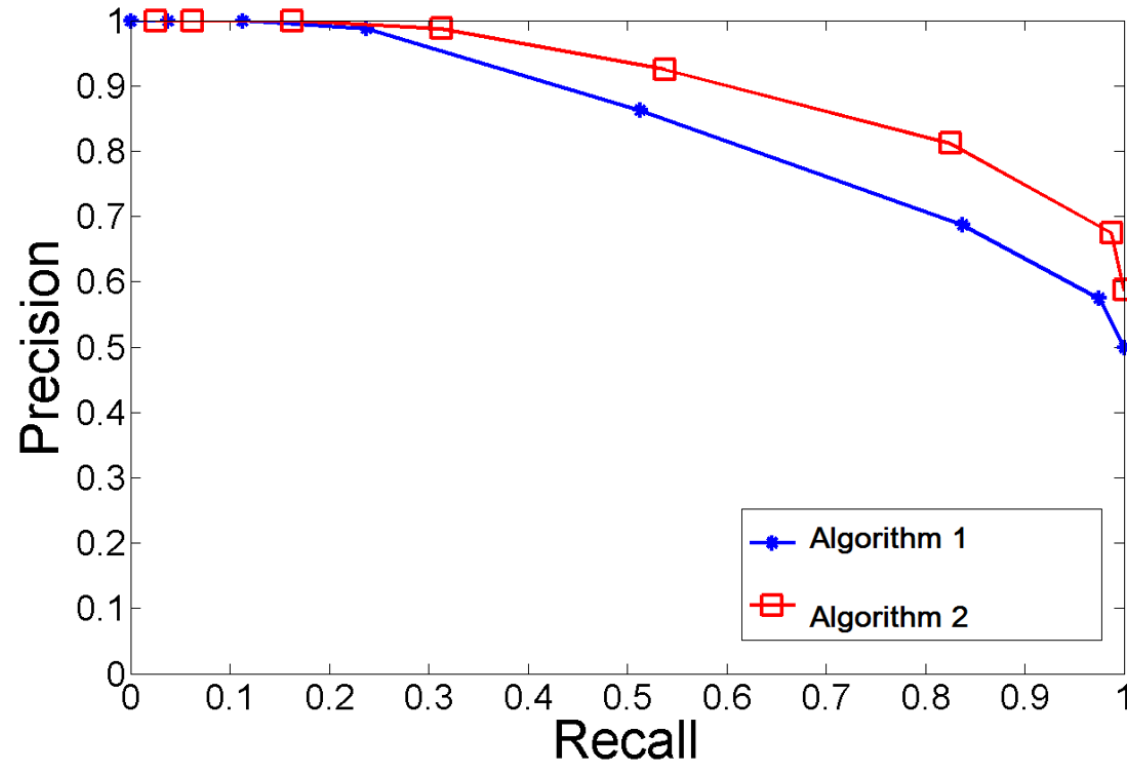
Thresholds for certainty



Common approach: assign class with highest probability

Alternative: require probability above a certain threshold

Visual aids to decide thresholds



Test different thresholds and plot the recall and precision that were achieved

Also common: Receiver Operator Characteristic (ROC) as curve of TPR vs FPR

Different but same for other techniques

- We considered classification
- Techniques may be different, problems remain the same
- Regression
 - Mean Squared Error (MSE)
 - Mean Absolute Percentage Error (MAPE)
 - R^2 coefficient
 - ...
- Rankings
 - Mean Average Precision in first K results (MAP@K)
 - Mean Reciprocal Rank (MRR)
 - Coverage
 - Personalization
 - ...
- NLP
 - ...

Which metric?

What does this mean for my use case?

What does this mean for my model?

What is a good baseline for comparison?

Evaluating generalization

Once Upon a Time ...



Label: no tank



Label: tank

Researchers tried to identify tanks hidden in a forest...



... it worked during training ...



... but failed with new pictures ...



Learned concept: not sunny

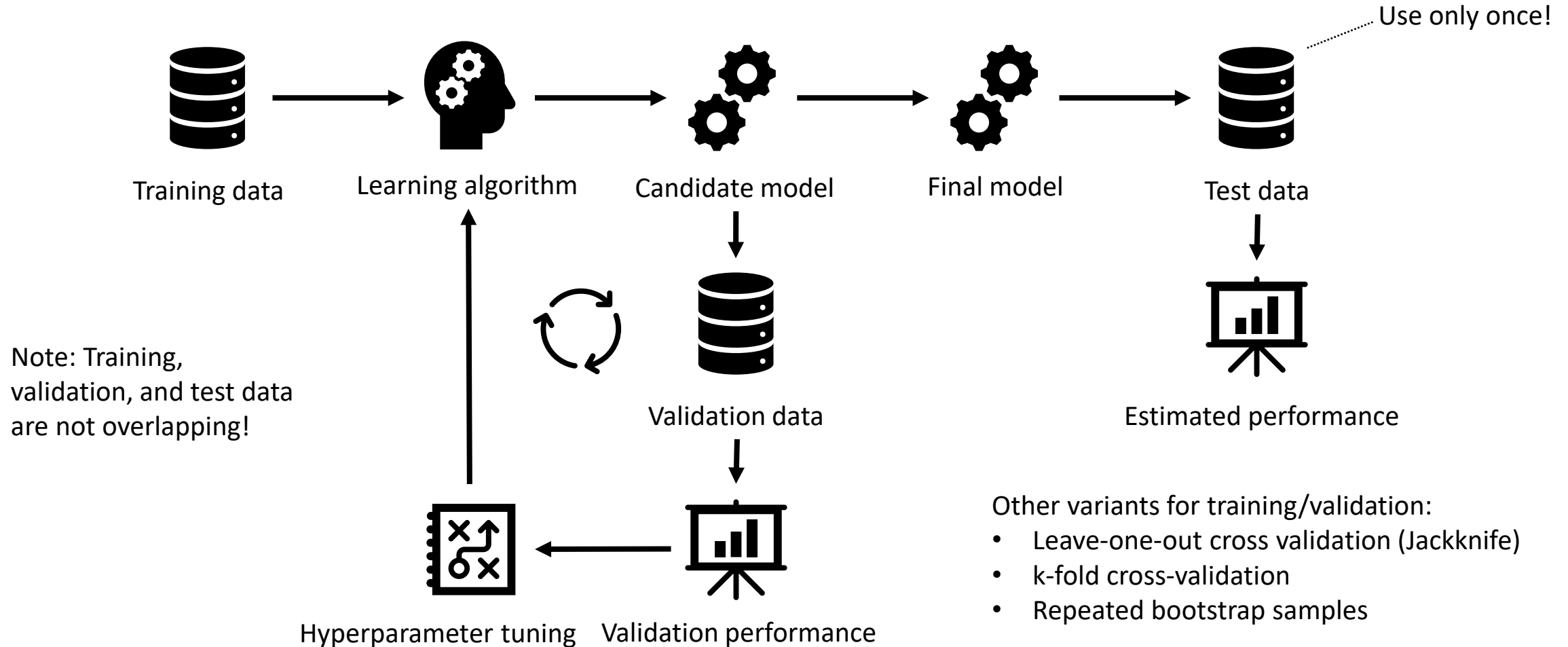


Learned concept: sunny

... because the algorithm learned something else!

This is an unconfirmed urban machine learning legend!

Training data, validation data, test data



Model parameters vs hyper parameters

- Model parameters

- The internal parameters that a model learns based on data
- Weights, coefficients, boolean conditions, ...
- Software engineering view:
 - Constants of the learned function



Optimized on training data

- Hyper parameters

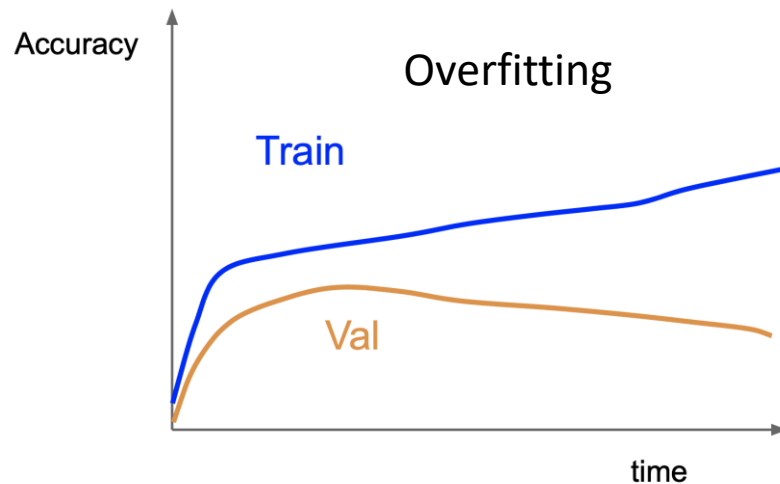
- The parameters that configure the algorithm that learns the model
- Number of neurons, layers, tree depth, ...
- Software engineering view:
 - Arguments of the learning algorithms interface



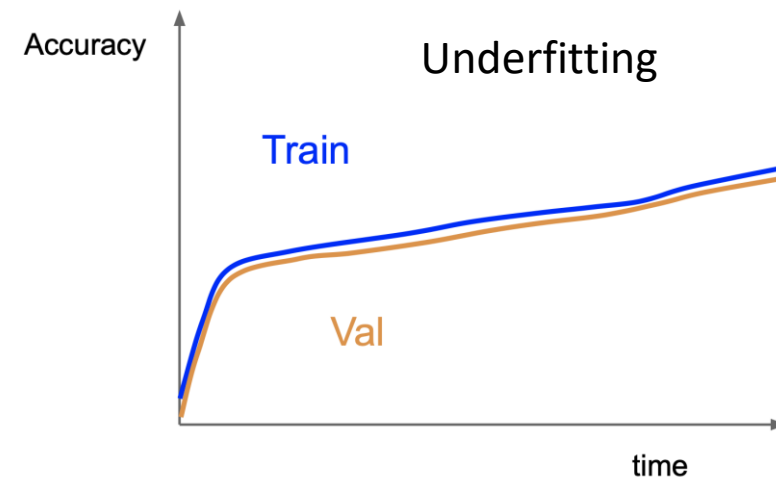
Optimized on validation data

Overfitting and underfitting

- Overfitting
 - Model learned exactly for the input data, but does not generalize to unseen data
 - Example: exact memorization
- Underfitting
 - Model makes very general observations that work on training data but poorly fits to other data
 - Example: Pictures with tanks are brighter



Memorizes training data (gets better)
Stops generalizing to other data (gets worse)



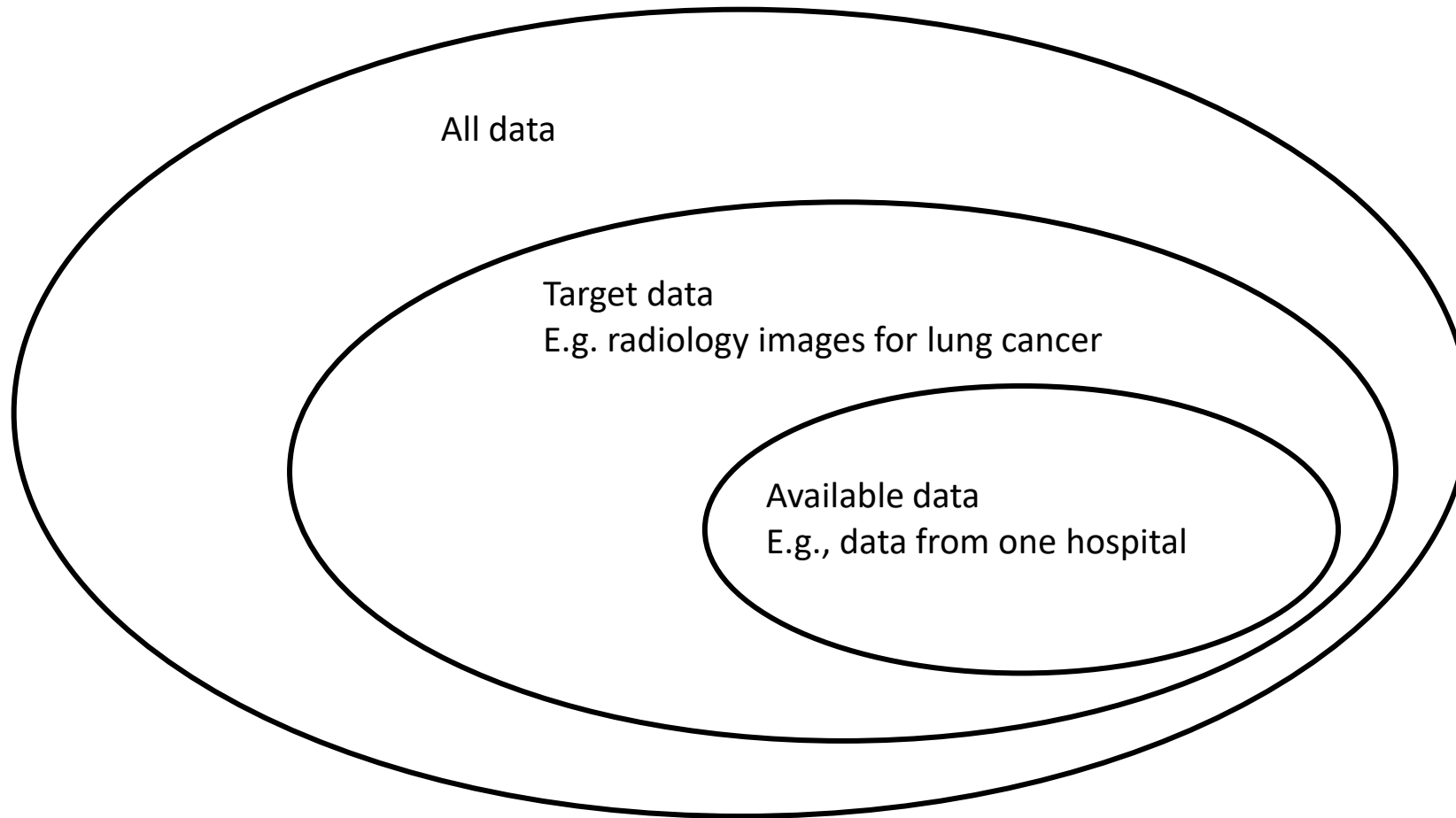
Model works equally (bad) on training and test data

Live exercise

- What can you do if your model overfits?
- What can you do if your model underfits?

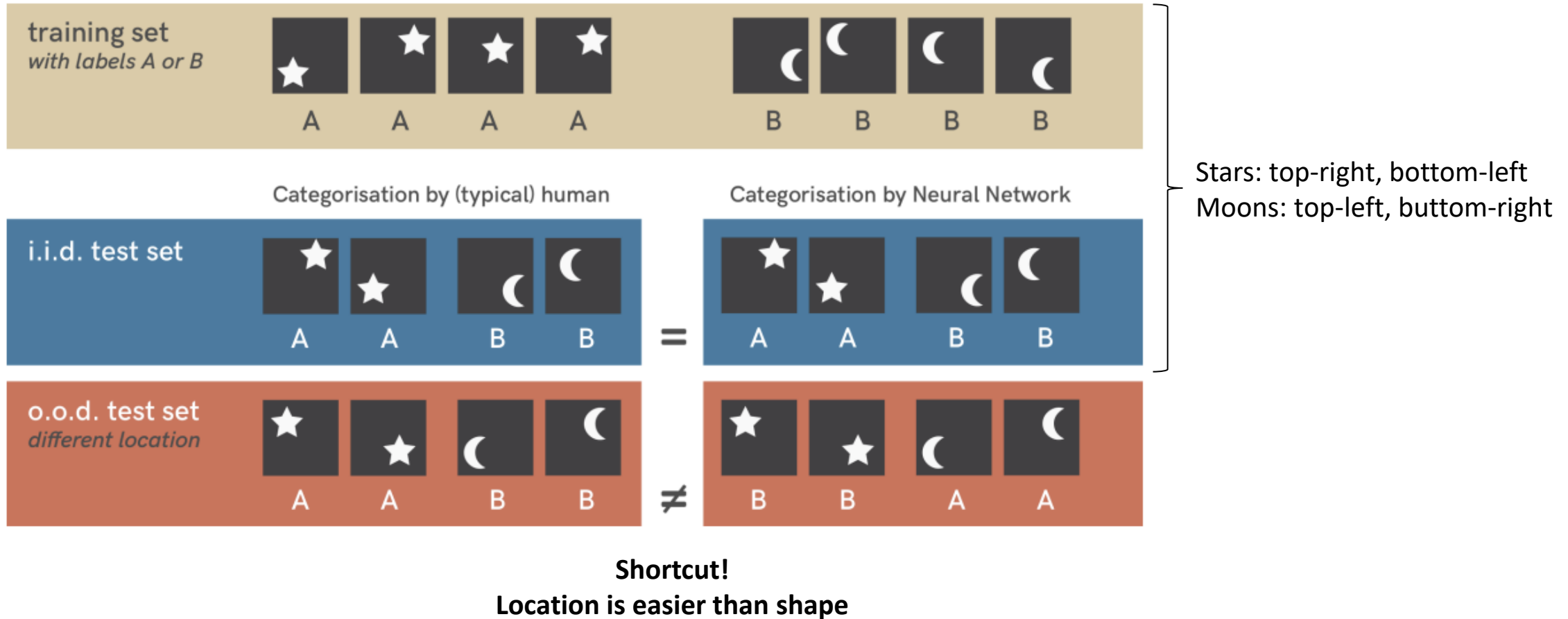
Pitfalls of evaluations

Test data not representative



Live exercise: What are reasons why the available data for the cancer prediction use case may not be representative?

Shortcut learning



Problems with representativeness in practice

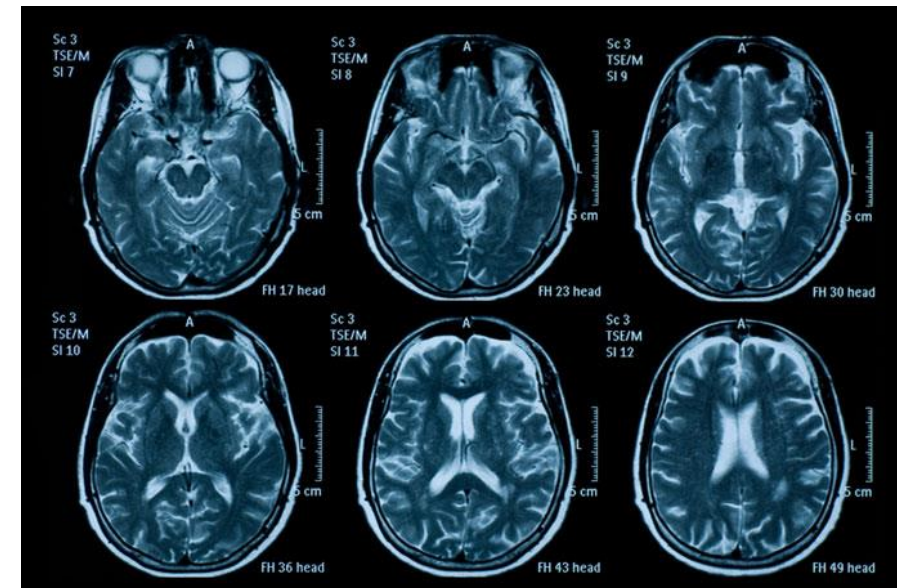
- Target distribution may not be known in early stages of the project
- Production data is the best test data
 - Requires knowledge about what production looks like
 - Sometimes only available after deploying initial system!
- Target distribution may shift over time
 - Concept drift!

Monitoring and continuous data collection are important!

Label leakage in images



Label leaked by sunlight



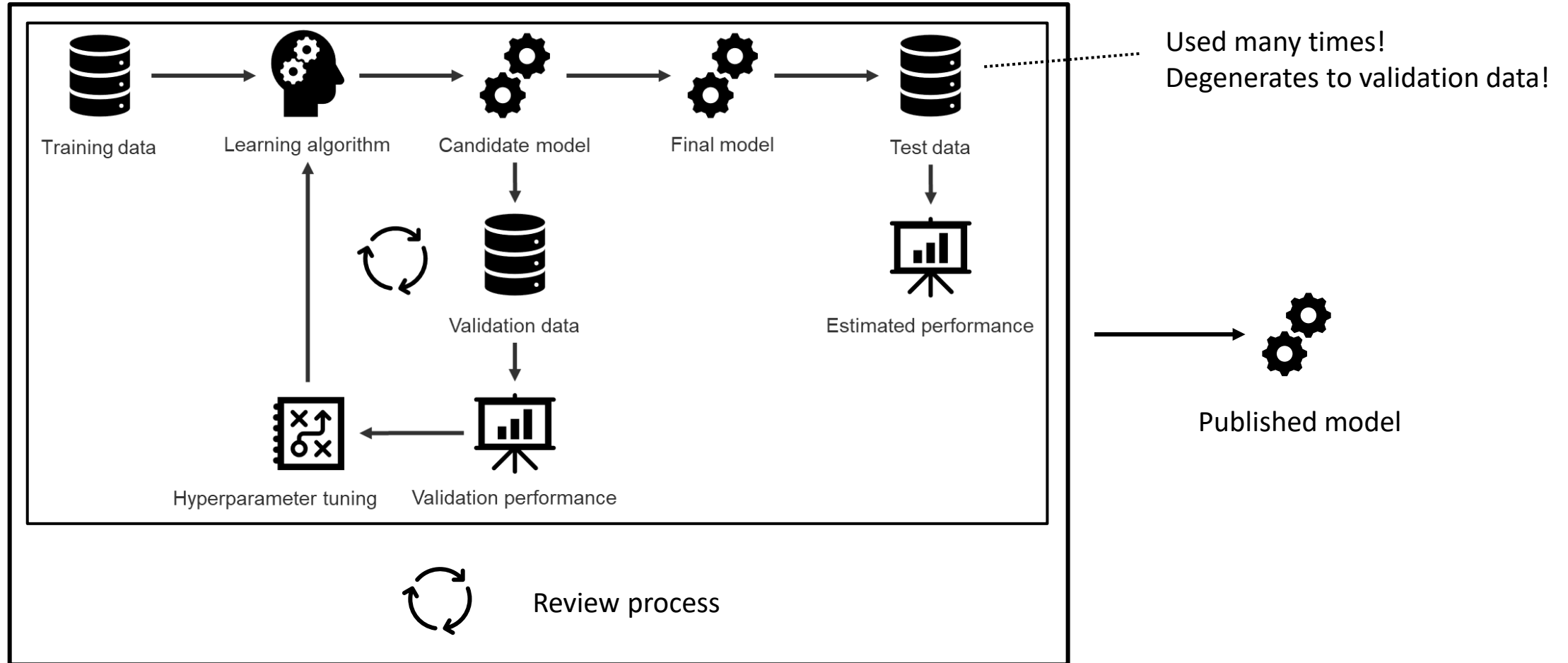
Label often leaked by metadata

Label leakage


- Label (or close correlates) included in inputs
- Examples:
 - Input: “interview conducted” in turnover prediction encode human judgement
 - Input: “has bank account” associates with predicting whether somebody will open one
- Live exercise:
 - Is label leakage always bad?

Be cautious of results that are too good to be true!

Overfitting on benchmarks



Overfitting in continuous experimentation systems




[Github](#) [Docs](#)

Listing Price Prediction

Experiment ID: 0 Artifact Location: /Users/matei/mlflow/demo/mlruns/0

Search Runs:

Filter Params: Filter Metrics:

4 matching runs 

	Time	User	Source	Version	Parameters		Metrics		
					alpha	l1_ratio	MAE	R2	RMSE
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.2	84.27	0.277	158.1
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.2	0.5	84.08	0.264	159.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.5	84.12	0.272	158.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0	0	84.49	0.249	161.2

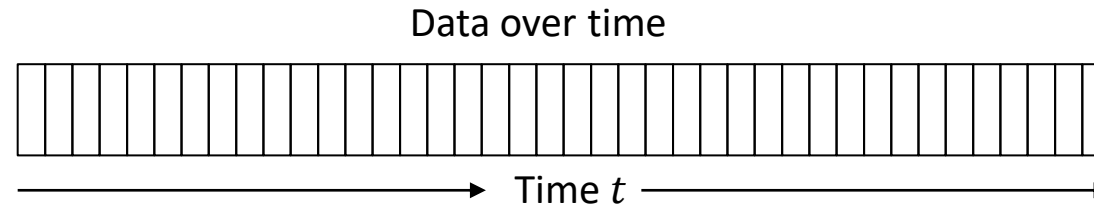
Part of MLOps to monitor performance of different models before serving

Test data needs to be changed often!

Degenerates to validation data otherwise!

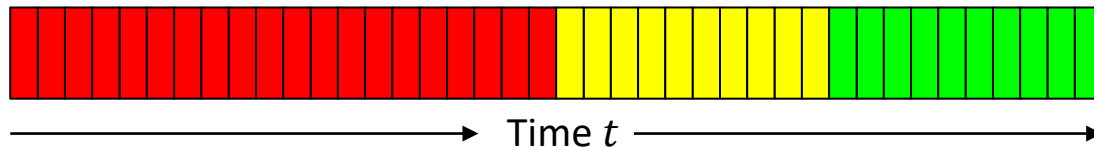
Use statistics to estimate when new data needs to be used.

Ignoring temporal dependencies

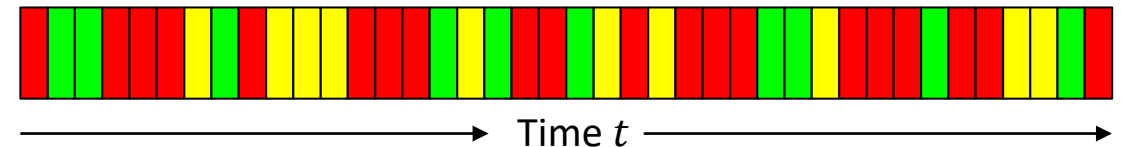


How can we split this into training, validation, and test data?

Good



Bad



Future predicts past!

Related datapoints



- Can be subtle issues!
 - Pictures of the same person (as above)
 - Sales from the same user
 - Pictures taken on the same day

Using misleading quality measures

- Using accuracy when costs of errors are different for false positives and false negatives
- Using the area under a curve of all possible thresholds, instead of only considering relevant thresholds
- Using global averages when risks and costs are different for subgroups of data
- Using old test data to calculate metrics
- Using very small validation and test sets
- Reporting results without suitable baselines
- Considering only gains (e.g., in recall), without considering related losses (e.g., precision)
- ...

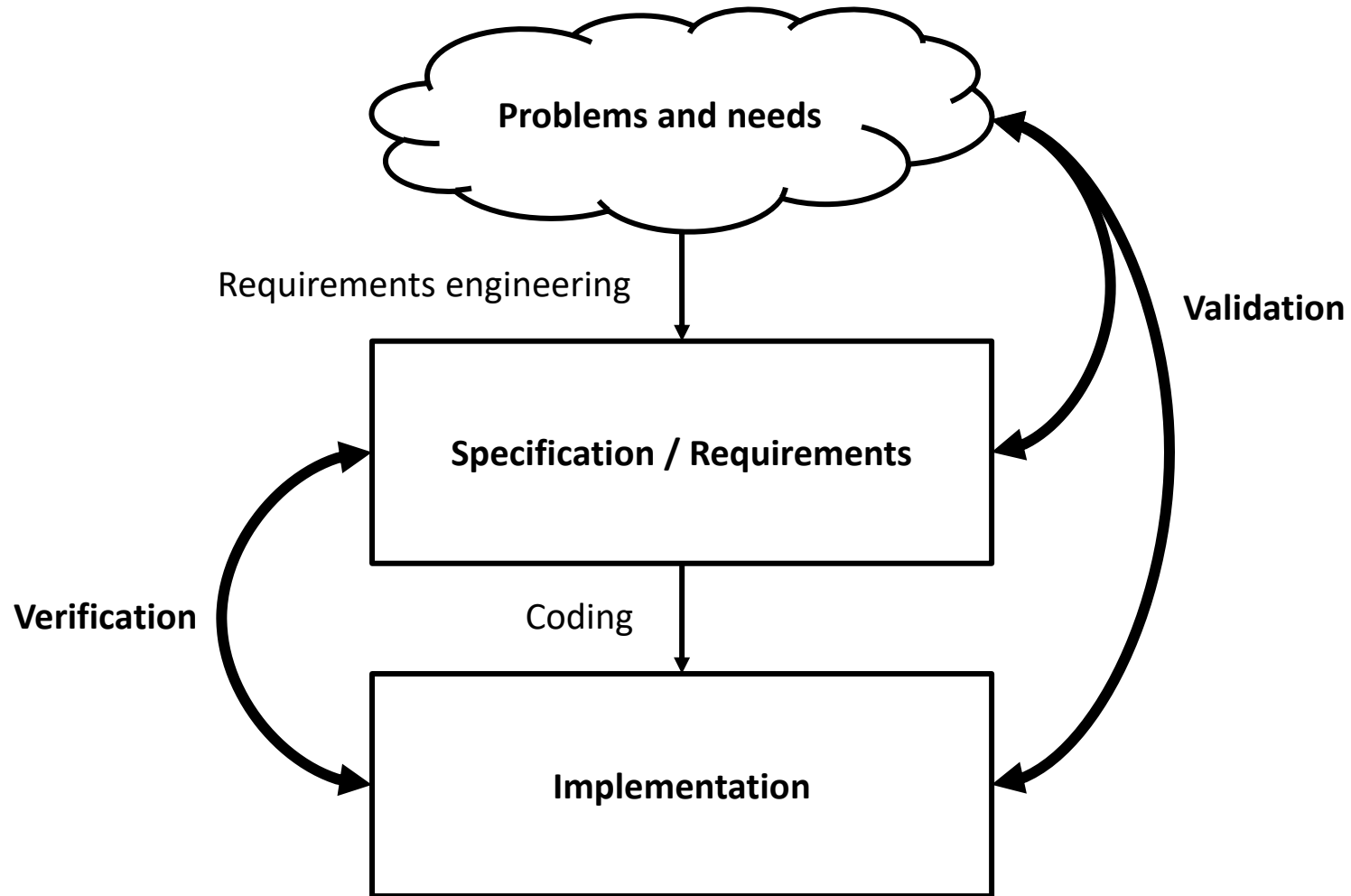
Correctness from a software engineering perspective

Functional correctness

```
/**
 * 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
 *
 * adjusted gross income must be positive;
 * returned deductions are not negative.
 */
float computeDeductions(float agi, Expenses expenses) {
    ...
}
```

- Given specification, do outputs match input?
- Each mismatch is considered a bug and should be fixed
 - Not economic to fix every bug, depends on cost of fix and cost of bug!

Verification vs. validation



Validation problem: Correct but useless?

Specification implemented correctly ...



... but the specification was wrong



Created system is not what the user needs

Often: bad, missing, or wrong assumptions about how the system is used

Strict correctness assumption

- Specification determines which outputs are correct/wrong
 - Binary! Not pretty good, 95% accurate, or correct for 98% of the users
- Single wrong results is a bug



Application calculates taxes

You pay calculated amount

If it is wrong and too little, you are in trouble!

Bug!

Ideally: Formal specification

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

Example: Specifying the behavior with Java Modeling Language (JML)

In practice: Informal and incomplete specifications

```
/**  
 * determines the largest value  
 */  
public static int max(int[] a);
```

Example: natural language comment specifies behavior

Live exercise: Why is this specification incomplete?

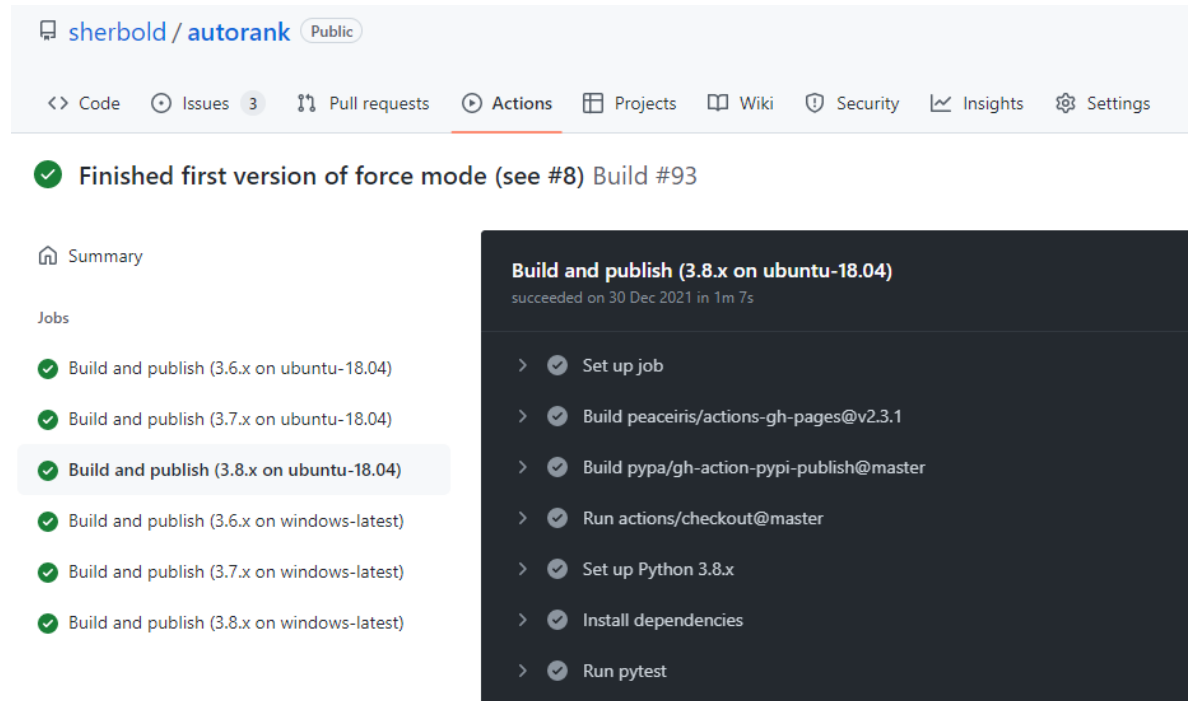
Common practices: Testing, automation, continuous integration

```
/**
 *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));
}
```

Write automated unit tests



The screenshot shows the GitHub Actions interface for the repository 'sherbold / autorank'. The 'Actions' tab is selected, displaying a workflow named 'Finished first version of force mode (see #8) Build #93'. A summary section lists the jobs, with 'Build and publish (3.8.x on ubuntu-18.04)' highlighted. A detailed view of this job shows a list of steps, all of which are completed successfully.

sherbold / autorank Public

<> Code Issues 3 Pull requests Actions Projects Wiki Security Insights Settings

✓ Finished first version of force mode (see #8) Build #93

Summary

Jobs

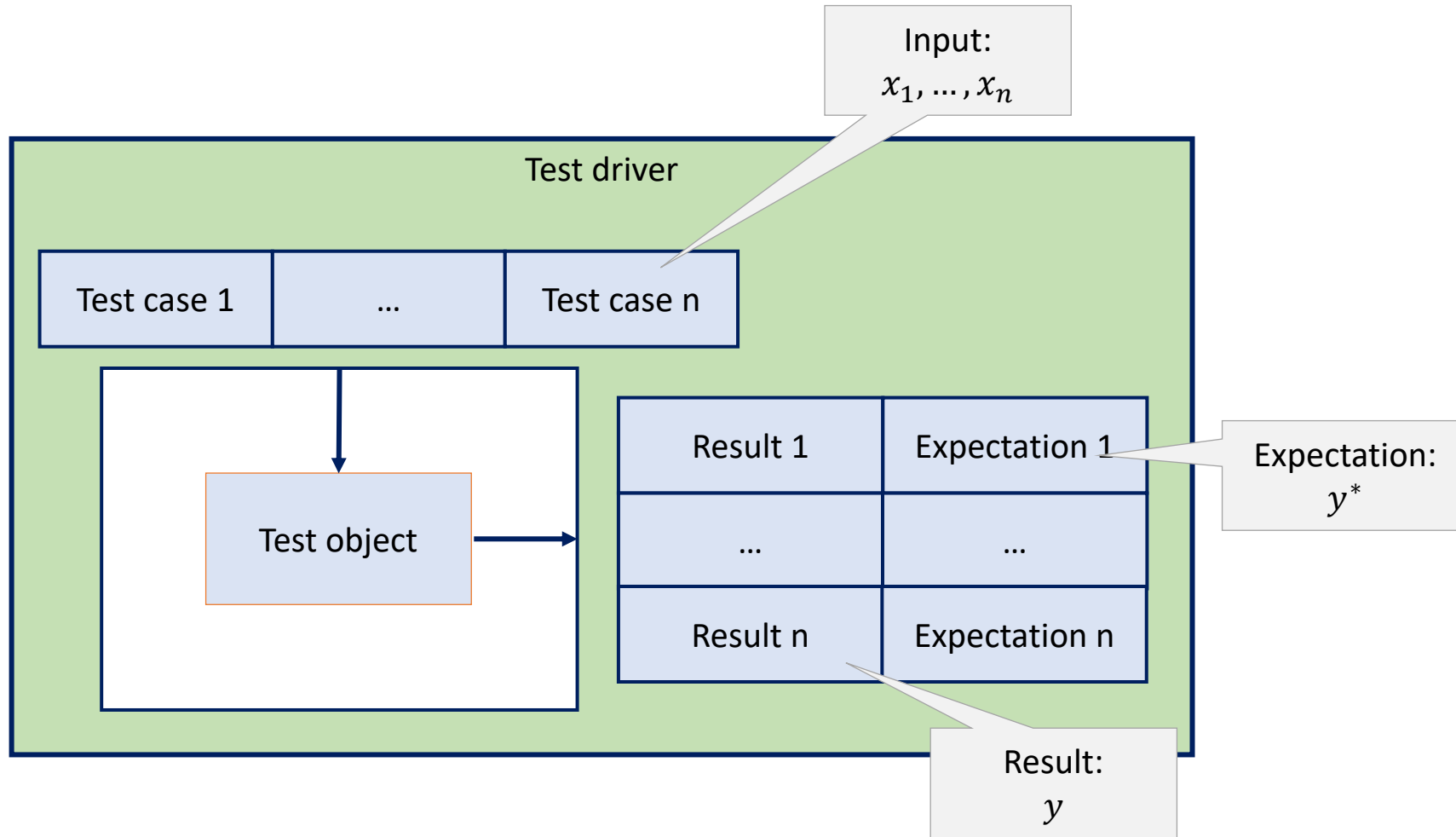
- ✓ Build and publish (3.6.x on ubuntu-18.04)
- ✓ Build and publish (3.7.x on ubuntu-18.04)
- ✓ Build and publish (3.8.x on ubuntu-18.04)
- ✓ Build and publish (3.6.x on windows-latest)
- ✓ Build and publish (3.7.x on windows-latest)
- ✓ Build and publish (3.8.x on windows-latest)

Build and publish (3.8.x on ubuntu-18.04)
succeeded on 30 Dec 2021 in 1m 7s

- > ✓ Set up job
- > ✓ Build peaceiris/actions-gh-pages@v2.3.1
- > ✓ Build pypa/gh-action-pypi-publish@master
- > ✓ Run actions/checkout@master
- > ✓ Set up Python 3.8.x
- > ✓ Install dependencies
- > ✓ Run pytest

Execute them automatically for each change

Basic software testing terminology

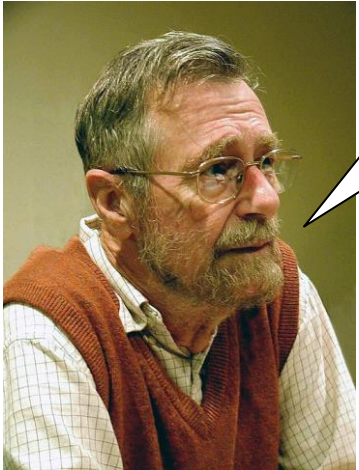


Expectation is determined by the test oracle.

Test oracle is (usually) derived from the specification

Important but imperfect

Testing shows the presence,
not the absence of bugs



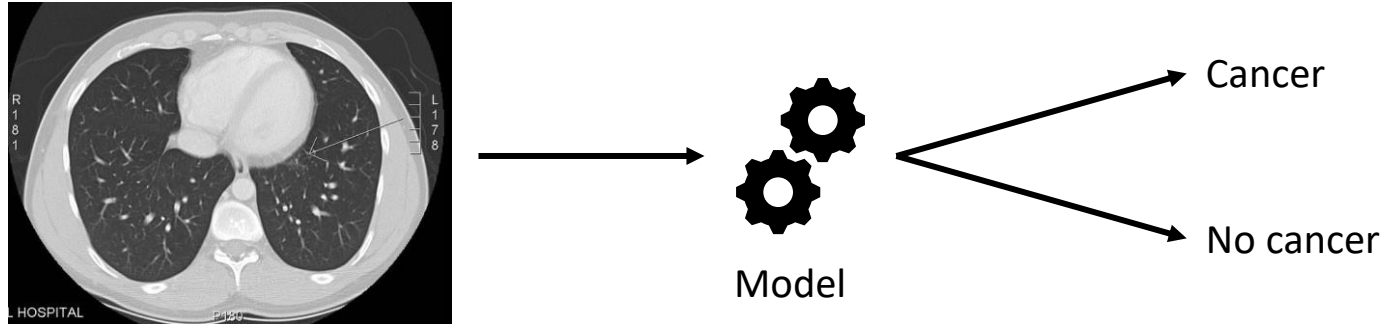
Edsger W. Dijkstra 1969

Many different quality aspects can be tested:

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

Correctness of machine learning from a software engineering perspective

Verification and validation of prediction tasks



```
/**  
 * returns whether the image indicates that the patient has cancer  
 */  
boolean hasCancer(Image scan);
```

How do we know the expected output?!

... no clear specification, rules unknown, instead only assumptions on expected behavior

Naive approach: Just write some unit tests

```
/**
 * returns whether the image indicates that the patient has cancer
 */
boolean hasCancer(Image scan);

@Test
void testPatient1() {
    assertFalse(hasCancer(loadImage("patient1.jpg")));
}

@Test
void testPatient2() {
    assertTrue(hasCancer(loadImage("patient2.jpg")));
}
```

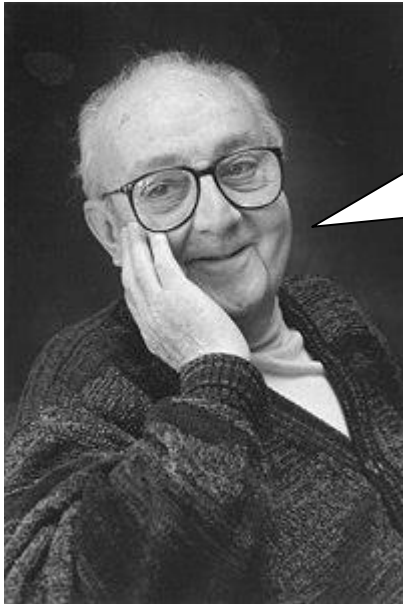
What does a test failure mean!?

Unless we expect 100% accuracy, some test failures are expected!

Weak correctness assumption

- Often no reliable ground truth
 - Human judgement may be biased
 - Humans may disagree
 - Problems may even have multiple acceptable solutions
- We must accept that mistakes happen
- We should still minimize the mistakes
- However, we must consider economics and feasibility
 - Maybe further improvement not possible (without a huge budget and years of development)
 - Maybe system already good enough
- We can expect to be better, the more similar our production data is to our training data

All models are wrong, but some models are useful!



George Box

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?"

Also the case beyond ML

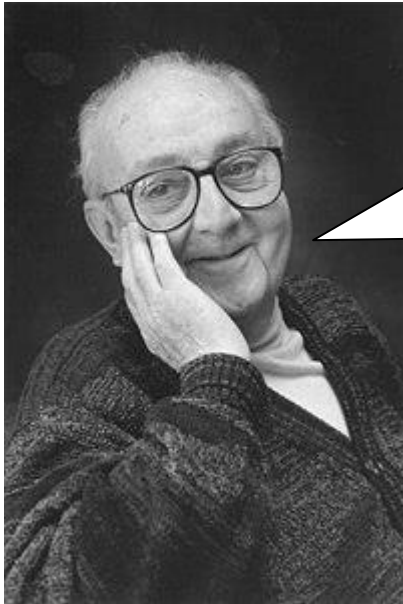
Newton's law of universal gravitation: $F = G \frac{m_1 m_2}{r^2}$

Very good model for normal situations

Inaccurate for extreme situations, e.g., very small scales, very high speeds, or very strong gravitation



All models are wrong, so we must be aware of it

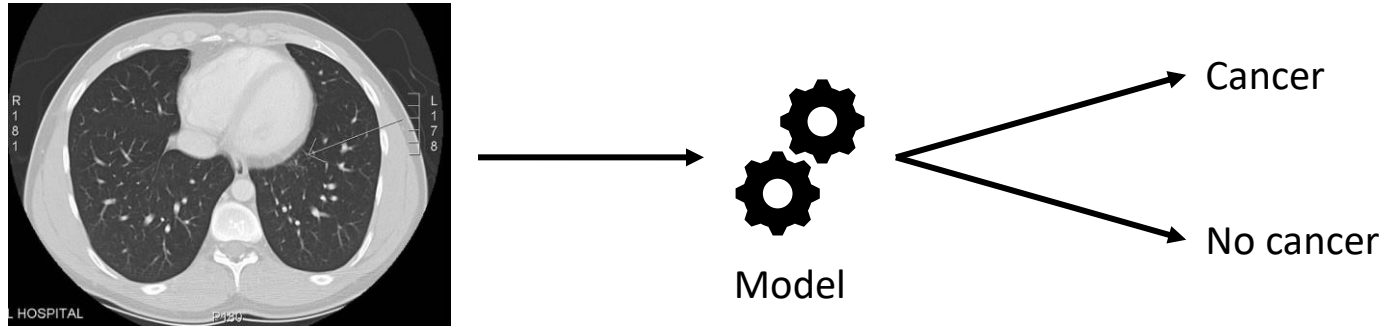


George Box

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.

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.

Models must have utility, to be useful, even if they are wrong



Can sometimes be wrong

... but not too often

... especially not when missing cancer!

} Similar to accuracy!

False negative: no treatment. False positive: more tests to confirm.

Machine learning models fit, or not

- A model is learned for given data with a given procedure
 - The learning process is typically ignored when considering model correctness
 - The model is generated, implementation issues are typically ignored
- Shifts correctness concerns from implementation to other issues!
- Is the data representative? Is there enough data? Is the data quality sufficient?
- Does the model learn meaningful concepts?
- Is the model useful for the purpose? Does it fit the data?
- Do model predictions usually fit users' expectations?
- Is the model consistent with other requirements?
 - E.g., fairness, robustness, privacy, ...

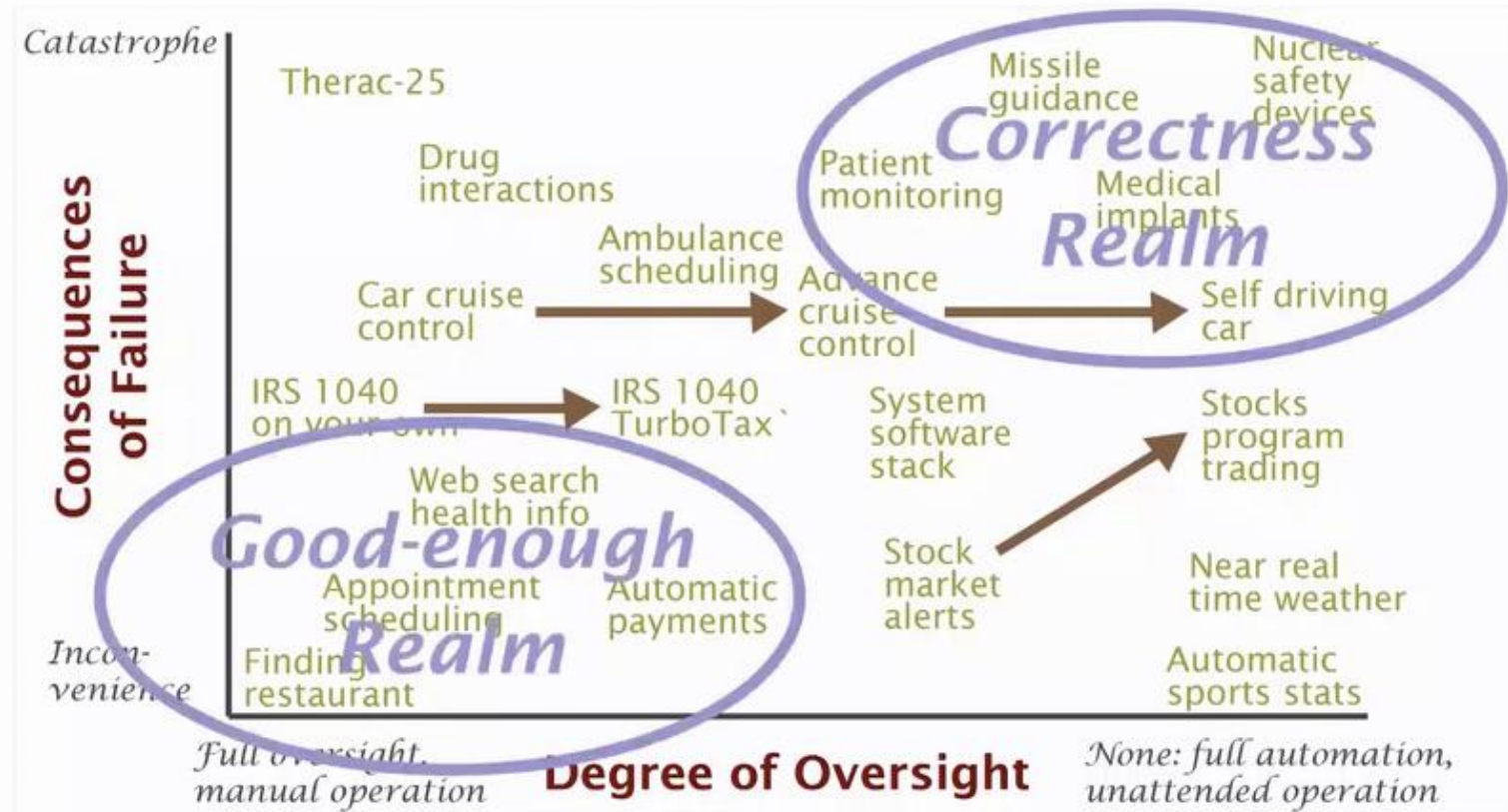
The typical software engineering view on correctness

- Software engineers almost never assure correctness
 - Testing finds bugs, but does not assure correctness
- Formal verification can assure correctness
 - Expensive and restricted to high-stakes applications
- Interaction between components and the environment often at least as much of a challenge
 - Hard to specify, many options

Good enough typical for software engineering!

Fit for purpose is software engineering goal, not correctness!

Required model quality depends on application



Mary Shaw

Some terminology

- Avoid the term *model bug*
 - The meaning of this term is unclear and it may mean multiple things, e.g.
 - An incorrect model
 - A correct model that has a bug in its implementation (e.g., bad numerics)
 - Any other sort of misbehavior that can be attributed to the model
- *Performance, accuracy, goodness of fit* (or just *fit*) are usually better terms to describe model quality than correctness
- *Testing* has multiple meanings, i.e., software testing vs. determination of model performance
 - Must be obvious from the context or clarified
- *Verification vs. validation* helps to guide process, but terms are often mixed up
 - May require longer explanations of the terms to avoid misunderstandings

Learning from software testing

Writing tests

- Simple function:

```
def nextDate(year: int, month: int, day: int):  
    ...
```

- Possible tests:

```
assert nextDate(2021, 2, 8) == (2021, 2, 9);  
assert nextDate(2021, 2, 28) == (2022, 3, 1);  
...
```

- Live exercise:
 - How do we define and select good tests?
 - How many tests do we need to feel confident that the function is not broken?

Software test case design

- Opportunistic/exploratory testing
 - Add some (unit) tests without much planning
- Specification-based testing (black box testing)
 - Derive test cases from specification
 - Equivalence classes, boundary value analysis, combinatorial testing, random testing, ...
- Structural testing (white box testing)
 - Derive test cases to cover implementation paths
 - Line coverage, branch coverage, data flow testing, modified condition/decision coverage, ...
- Test execution ideally automated, but can be manual as well
- Automated generation from specification or code possible
 - Depends on type of tests and formality of code/specification

Example for deriving tests

```
def nextDate(year: int, month: int, day: int):  
    ...
```

- Equivalence classes for day:
 - $day \in \{1, \dots, 27\}$
 - $day = 28$
 - $day = 29$
 - $day = 30$
 - $day = 31$
- Equivalence classes for month:
 - $month \in \{1, 3, 5, 7, 8, 10, 12\}$
 - $month \in \{4, 6, 9, 11\}$
 - $month = 2$
- Equivalence classes for year
 - $year \in \{leap\ years\}$
 - $year \in \{not\ leap\ years\}$

Equivalence class analysis

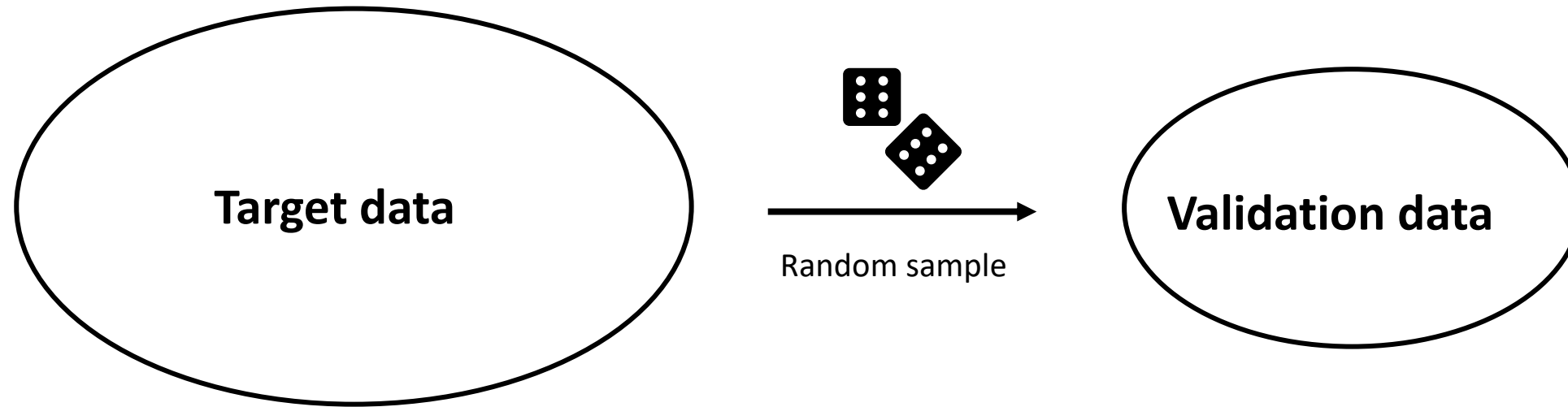
Pick one value from each group, combine groups from variables, determine output

Boundary value analysis

Pick minimum, minimum+1, some middle value, maximum-1 and maximum from each group

If reasonable include invalid values minimum-1 and maximum+1

Representativeness of validation data



Is this representative?

Is this suitable for software testing?

Not all inputs are equal



→ Racist equivalence classes?

There Is a Racial Divide in Speech-Recognition Systems, Researchers Say: Technology from Amazon, Apple, Google, IBM and Microsoft misidentified 35 percent of words from people who were black. White people fared much better.

<https://www.nytimes.com/2020/03/23/technology/speech-recognition-bias-apple-amazon-google.html>

Random mistakes vs. rare but biased mistake



Door camera automatically detects if somebody is there

... but only for people taller than 150 cm



Spam detection has almost no false positives

... but filters all real alerts from your bank

Consider separate evaluations for important subpopulations!

Monitor mistakes in production to uncover issues!

Identify important inputs

- Curate validation data for specific problems and subpopulations
- *Regression testing*
 - Validation data for important inputs
 - High accuracy, must not get worse over time
- *Uniformness/fairness testing*
 - Separate validation data for subpopulations (e.g., accents, types of emails)
 - All groups should achieve comparable accuracy
- *Setting goals*
 - Validation data with difficult cases that are optional to be handled correctly (e.g., correct spelling of all names)
 - Low accuracy acceptable
- All sets should be derived from the requirements on the system!
 - Specification-based testing!

Input partitioning

- Accuracy within subgroups (→ uniformness and fairness testing)
 - Can also be relevant for business objectives!
 - A customer that brings ten times the revenues is more important than the average customer
- Achieved by slicing data based on population criteria
- Interactions between slices also need to be considered!
- Goal:
 - Identify problems
 - Plan mitigations
- Mitigation may also be accepting (and reporting!) lower confidence in results for a subgroup

Live exercise:

Input partitioning for cancer prediction

- What would be relevant slices?
- What would you expect/require for the slices?



Input partitioning example: Movie review prediction

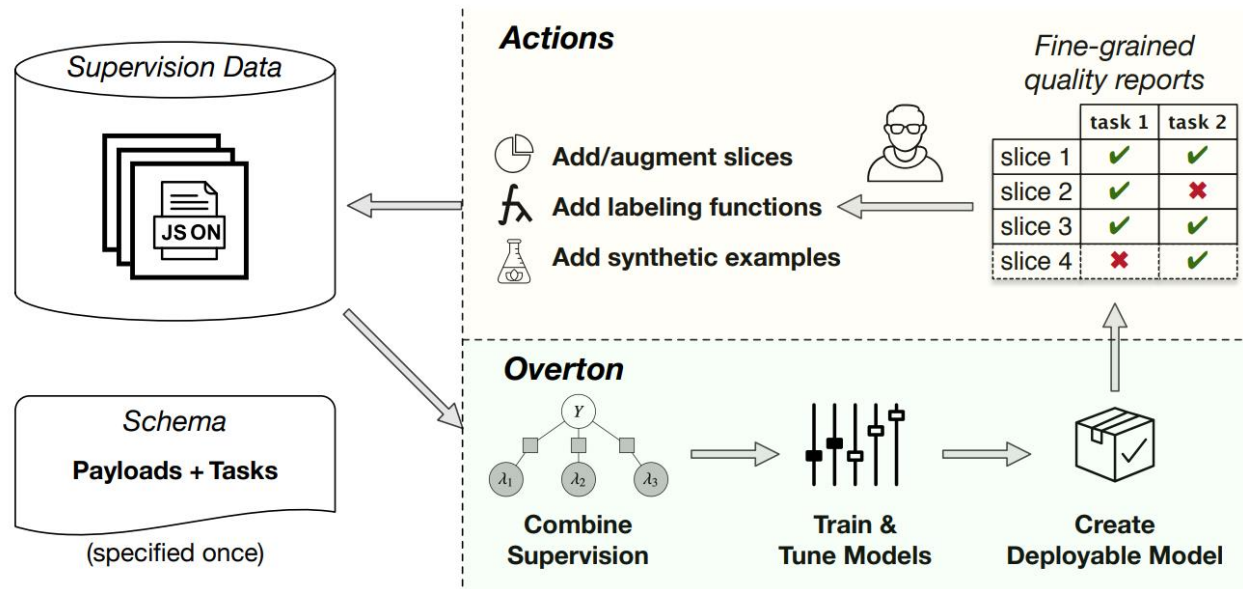
DECADE	Support	Accuracy
1910s	38	78.94
1930s	338	87.87
1990s	3007	90.95
2000s	6192	91.40

Low accuracy, but also only a small amount of data (support) for old movies

MAIN_GENRE	RAT_CAT	LEN_CAT	Support	Accuracy
Mystery	OK	Long	11	72.72
Fantasy	OK	Short	36	77.77
Crime	OK	Long	100	81.00
Comedy	GOOD	Long	55	96.36

Accuracy and support for more complex slices also differ

Model improvement at Apple with Overton



Human focus on data and slices

AutoML approach for creating model

Testing model capabilities

- Check if model can represent abstract concepts
- Examples for sentiment mining concepts
 - Negations
 - Robustness to typos
 - Dealing with synonyms and abbreviations
 - Ignoring irrelevant information
 - ...
- Create validation data for individual capabilities
 - Multiple examples for each capability
 - Manually or following patterns

Example: Robustness of sentiments

Evaluate if the model contains the expected invariants

Appending randomly generated handles or URLs should not affect sentiment

@JetBlue that selfie was extreme @askdj
@united stuck because staff took a break? Not happy ... <https://t.co/eqwe>

Small typos through swapping characters should not affect sentiment

@JetBlue@JeBtlue that selfie was extreme @askdj
@SouthwestAir no thanksthakns

Switching locations should not change sentiment

@JetBlue I want you guys to be first to fly to #CubaCanada
@VirginAmerica I miss the #nerdbird in SanJoseDenver

Switching person names should not change sentiment

... Airport agents were horrendous. SharonErin was our savior
@united 8602947, JonSean at <http://t.co/58tuTgli0D>, thanks.

Example: Negation capabilities

Evaluate if the model handles negation of words as expected

Negated negative should be positive or neutral

The food is not poor
It isn't a lousy customer service

Negated neutral should still be neutral

This aircraft is not private.
This is not an international flight.

Negation of negative at the end should be positive or neutral

I thought the plane would be awful, but it wasn't.
I thought I would dislike that plane, but I didn't.

Negated positive with neutral content in the middle should be negative

I wouldn't say, given it's a Tuesday, that this pilot was great.
I don't think, given my history with airplanes, that this is an amazing staff.

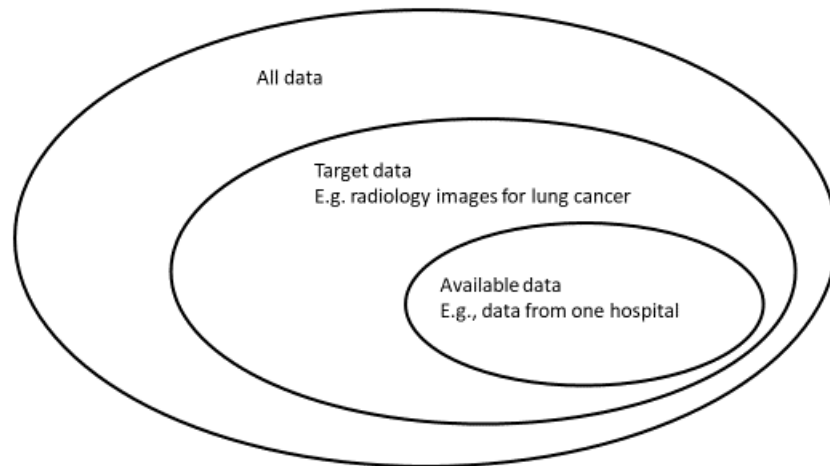
Live exercise: What are capabilities of the cancer classifier?

- Invariants?
- Expectations regarding predictions?



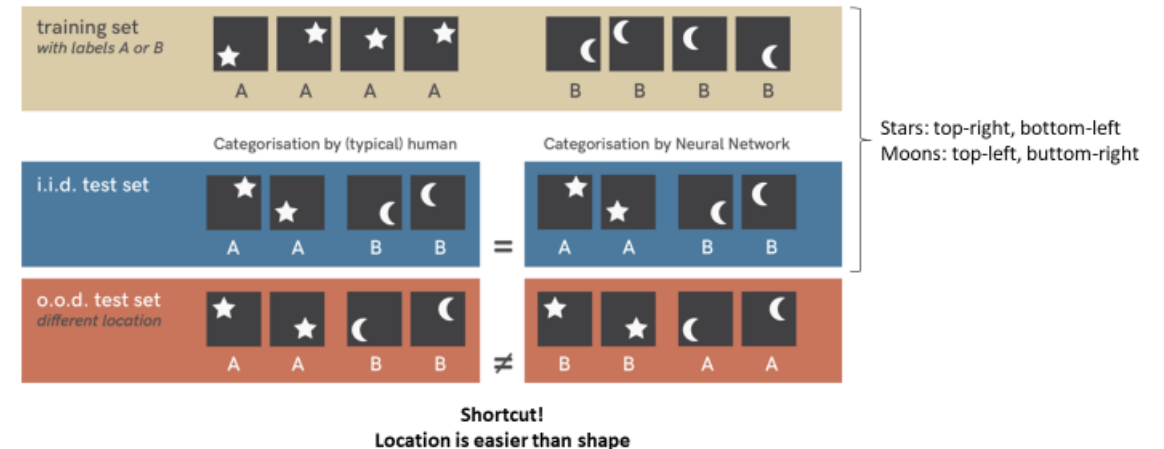
Usually not fair to expect generalization beyond training distribution

Test data not representative



Live exercise: What are reasons why the available data for the cancer prediction use case may not be representative?

Shortcut learning



Geirhos, Robert, Björn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. "Shortcut learning in deep neural networks." *Nature Machine Intelligence* 2, no. 11 (2020): 665-673.

Testing capabilities may help with generalization

- Capability \approx “partial specification” that is independent of training data
- Tests encode domain knowledge about the problem
 - Capabilities are inherently domain specific
 - Curating capability specific test data may be problematic / expensive
- Testing for capabilities helps identify shortcuts
 - Only works if the shortcut is not present in curated data!
- Helps to build trust in the model beyond training data

Strategies for identifying capabilities

- Analyze common mistakes
 - E.g., manually analyze past mistakes to find patterns
- Use existing knowledge about the problem
 - E.g., about natural language or the problem domain
- Observe humans at the task
 - E.g., consider the aspects that radiologists look for in MRIs
- Derive from fairness requirements
 - E.g., location should not have an effect on sentiment (bias)
- ...

Generating test data for capabilities: Domain-specific generators

I {NEGATION} {POSITIVE_VERB} the {THING}.

Uses curated negations, verbs, and things to generate many examples of negative statements.



(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan



(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat



(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Combines foreign texture with shape to test if the model focuses on shape or texture

Generating test data for capabilities: Mutating existing inputs

E.g., synonyms, distractions, noise

Example: Robustness of sentiments

Evaluate if the model contains the expected invariants

Appending randomly generated handles or URLs should not affect sentiment

@JetBlue that selfie was extreme @asikdj
@united stuck because staff took a break? Not happy ... <https://t.co/eqwe>

Small typos through swapping characters should not affect sentiment

@JetBlue@JeBtue that selfie was extreme @asikdj
@SouthwestAir no thanks thakns

Switching locations should not change sentiment

@JetBlue I want you guys to be first to fly to #CubaCanada
@VirginAmerica I miss the #nerdbird in San-JoseDenver

Switching person names should not change sentiment

... Airport agents were horrendous. SharonErin was our savior
@united 8602947, JonSean at <http://t.co/58tuTgli0D>, thanks.

Ribeiro, Marco Tulio, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. "Beyond Accuracy: Behavioral Testing of NLP Models with CheckList." In Proceedings ACL, p. 4902–4912. (2020).

Example: Robustness of sentiments

Evaluate if the model contains the expected invariants

Appending randomly generated handles or URLs should not affect sentiment

@JetBlue that selfie was extreme @asikdj
@united stuck because staff took a break? Not happy ... <https://t.co/eqwe>

Small typos through swapping characters should not affect sentiment

@JetBlue@JeBtue that selfie was extreme @asikdj
@SouthwestAir no thanks thakns

Switching locations should not change sentiment

@JetBlue I want you guys to be first to fly to #CubaCanada
@VirginAmerica I miss the #nerdbird in San-JoseDenver

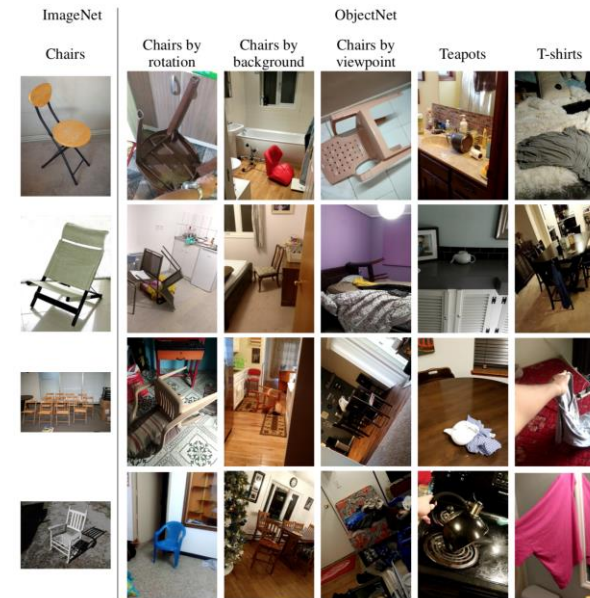
Switching person names should not change sentiment

... Airport agents were horrendous. SharonErin was our savior
@united 8602947, JonSean at <http://t.co/58tuTgli0D>, thanks.

Ribeiro, Marco Tulio, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. "Beyond Accuracy: Behavioral Testing of NLP Models with CheckList." In Proceedings ACL, p. 4902–4912. (2020).

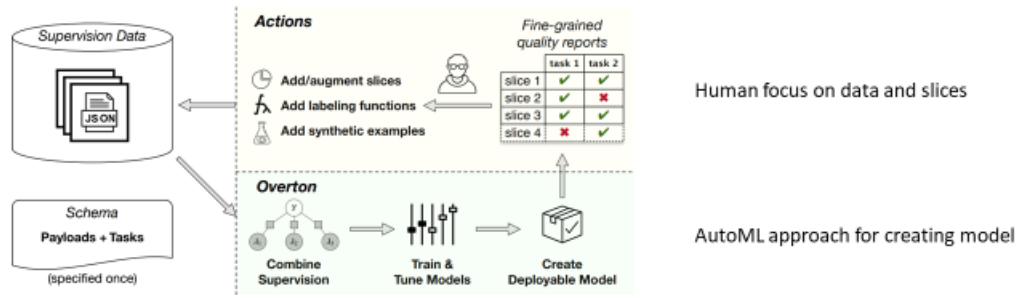
Generating test data for capabilities: Crowd-sourcing test generation

- Ask humans to modify data
- Example: minimally change sentence to flip the sentiment with sarcasm
 - The world of Atlantis, hidden beneath the earth's core, is fantastic!
 - The world of Atlantis, hidden beneath the earth's core, is "fantastic"!
- Ask humans to generate data



Generating test data for capabilities: Slicing test data

Model improvement at Apple with Overton



Christopher Ré, Feng Niu, Pallavi Gudipati, Charles Srisuwananukorn. "Overton: A Data System for Monitoring and Improving Machine-Learned Products." Proc. of the Conference on Innovative Data Systems Research, 2020.

Find slices of input containing "not"

Find slices of input that are very bright/dark

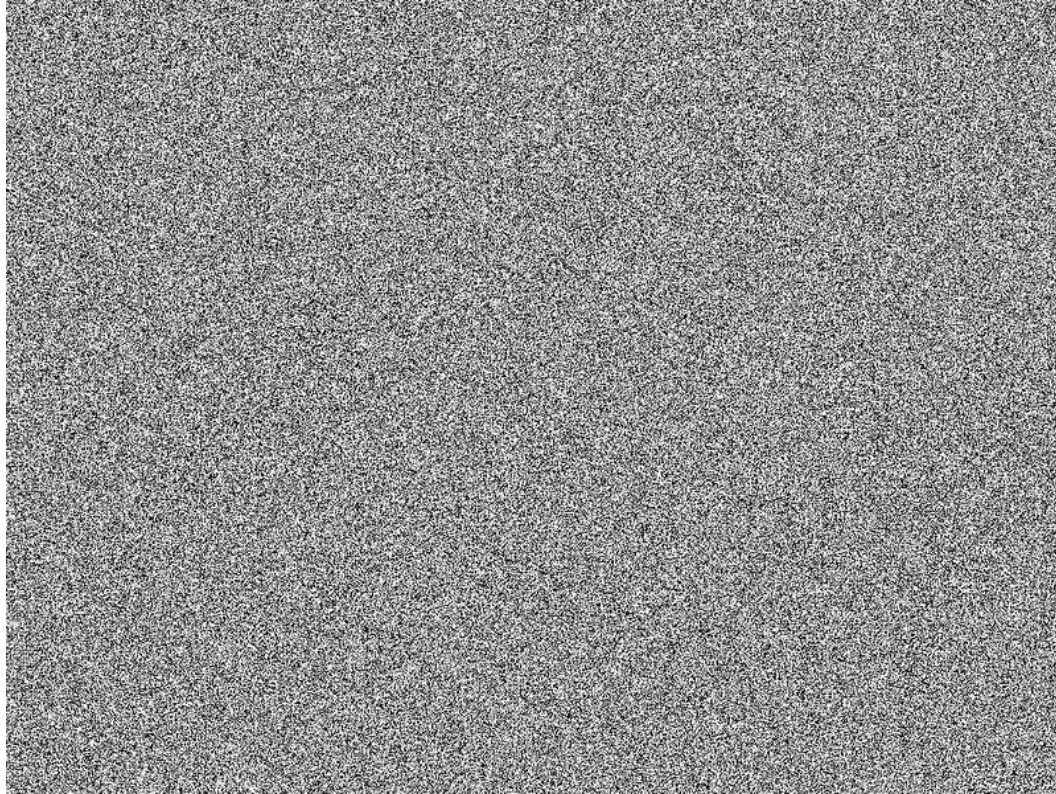
...

Testing vs. training capabilities

- Insights regarding capabilities relevant for both testing and training
- Capabilities that hold during testing should also hold during training
- If capabilities do not hold during testing, we can investigate why
 - Overfitting during training
 - Underfitting during training
 - Capability not (often) observed during training
 - ...
- Can curate data for training to get better!

The oracle problem

Random input generation is easy!



... but does it really make sense to test cancer prediction with this?!

Randomly generating realistic inputs often also possible

- Random dates
- Random texts
- Randomly generated images
- Randomly generated coordinates
- Randomly generated driving behavior
- ...

How do we know the expected output?!

The Oracle problem



In general, we have no way to determine the expected output for an input!

→ No test oracle!

Test case generation and test oracles

- Manual test oracle
 - Human constructs input-output pairs
- Checking of global properties only
 - Crashes, overflows, code injection, ...
- Comparison against a gold standard
 - Alternative implementation, executable specification
- Manually written assertions as runtime checks
 - Checks part of the specification at runtime

Manual oracle

```
@Test
void testNextDate() {
    assert nextDate(2010, 8, 20) == (2010, 8, 21);
    assert nextDate(2024, 7, 15) == (2024, 7, 16);
    assert nextDate(2011, 10, 27) == (2011, 10, 28);
    assert nextDate(2024, 5, 4) == (2024, 5, 5);
    assert nextDate(2013, 8, 27) == (2013, 8, 28);
    assert nextDate(2010, 2, 30) throws InvalidInputException;
}
```

```
@Test
void testCancerPrediction() {
    assert cancerModel.predict(loadImage("random1.jpg")) == true;
    assert cancerModel.predict(loadImage("random2.jpg")) == true;
    assert cancerModel.predict(loadImage("random3.jpg")) == false;
}
```



Does not (easily) scale, no automation

Global properties

```
@Test
void testNextDate() {
    nextDate(2010, 8, 20)
    nextDate(2024, 7, 15)
    nextDate(2011, 10, 27)
    nextDate(2024, 5, 4)
    nextDate(2013, 8, 27)
    nextDate(2010, 2, 30)
}
```

```
@Test
void testCancerPrediction() {
    cancerModel.predict(generateRandomImage())
    cancerModel.predict(generateRandomImage())
    cancerModel.predict(generateRandomImage())
}
```

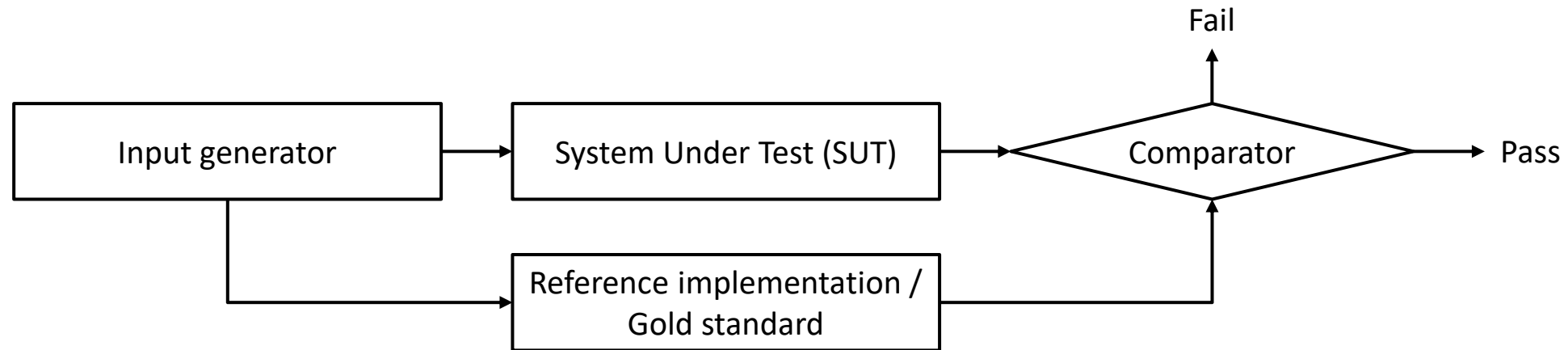
Only fails in case of unhandled exceptions

Effective to find severe issues!

... but ignores correctness of computed results



Comparison against reference implementation



```
@Test
void testNextDate() {
    assert nextDate(2010, 8, 20) == referenceLib.nextDate(2010, 8, 20);
    assert nextDate(2024, 7, 15) == referenceLib.nextDate(2024, 7, 15);
    assert nextDate(2011, 10, 27) == referenceLib.nextDate(2011, 10, 27);
    assert nextDate(2024, 5, 4) == referenceLib.nextDate(2024, 5, 4);
    assert nextDate(2013, 8, 27) == referenceLib.nextDate(2013, 8, 27);
    assert nextDate(2010, 2, 30) == referenceLib.nextDate(2010, 2, 30)
}
```

Assumes we have a correct reference implementation

```
@Test
void testCancerPrediction() {
    assert cancerModel.predict(loadImage("random1.jpg")) == ???;
}
```

... which is usually not the case for ML problems

Checking invariants at runtime

```
class Stack {  
    int size = 0;  
    int MAX_SIZE = 100;  
    String[] data = new String[MAX_SIZE];  
    // class invariant checked before and after every method  
    private void check() {  
        assert(size >= 0 && size <= MAX_SIZE);  
    }  
    public void push(String v) {  
        check();  
        if (size < MAX_SIZE)  
            data[size++] = v;  
        check();  
    }  
    public void pop(String v) { check(); ... }  
}
```

Assumes we can specify invariants within model execution

... unclear how this should work for ML in general

```
@Test  
void testStackRandom() {  
    //randomly generated sequence of calls  
    Stack s = new Stack();  
    s.push("foo");  
    s.push("bar");  
    s.pop();  
    ...  
}
```

ML Models == Untestable Software?!

- Manual test oracle
 - Does not scale
- Checking of global properties only
 - Only suitable for a severe bugs
- Comparison against a gold standard
 - Usually, no model that is always correct available
- Manually written assertions as runtime checks
 - Unclear how to define or insert into code



Pseudo-oracles



Simulate real test oracles for a subset of the input-output pairs

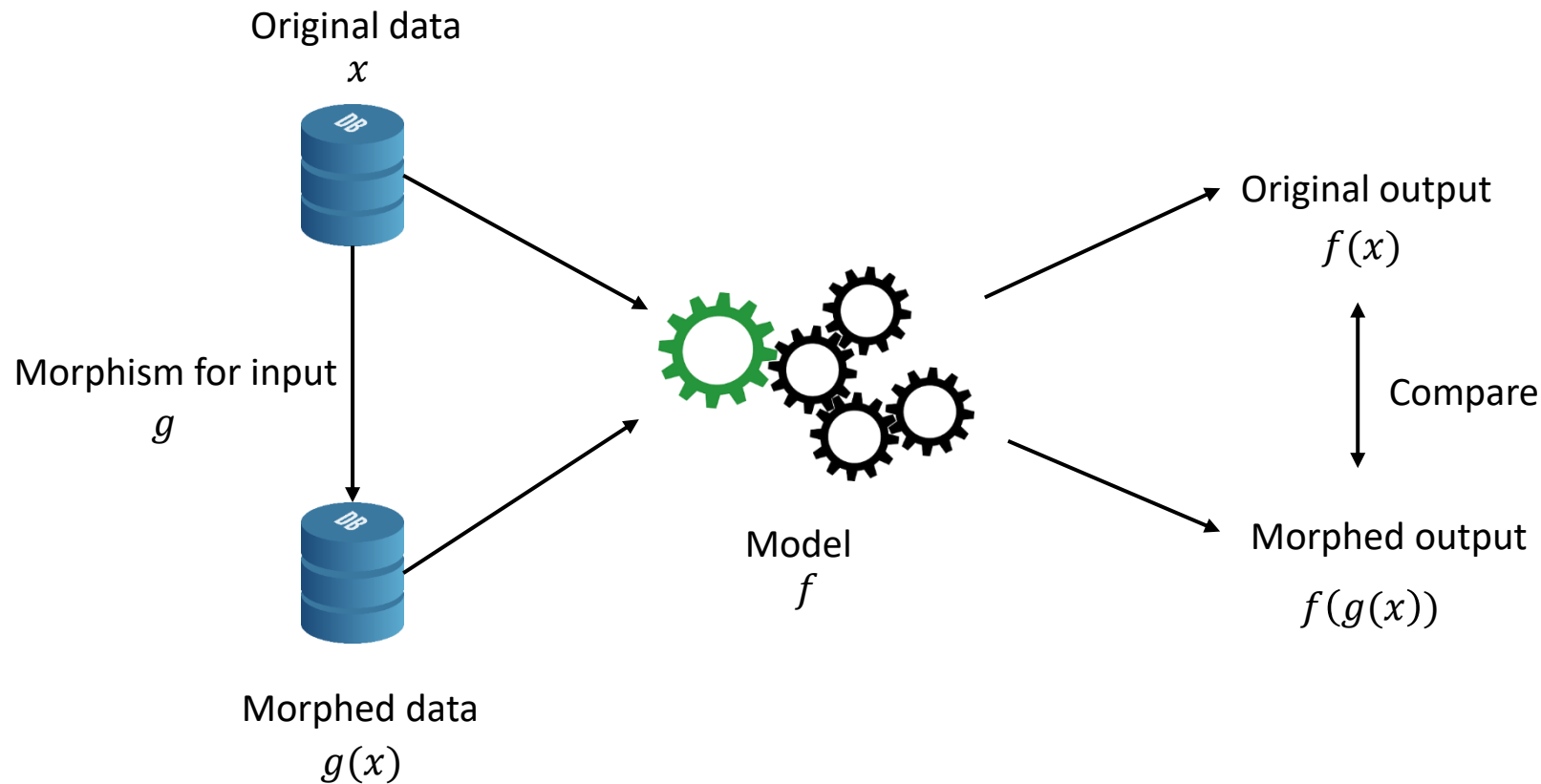
Cannot be used to test everything, but useful for testing specific properties!

Pseudo-oracles with metamorphic testing

Metamorphic testing for machine learning

Define transformation (morphism) g for input to specify invariant

Check invariant relationship between $f(x)$ and $f(g(x))$ as test oracle



We have already used this!

Generating test data for capabilities: Mutating existing inputs

E.g., synonyms, distractions, noise

Example: Robustness of sentiments

Evaluate if the model contains the expected invariants

Appending randomly generated handles or URLs should not affect sentiment

@testBlue that selfie was extreme @asidj
@united stuck because staff took a break? Not happy - https://t.co/vqjve

Small typos through swapping characters should not affect sentiment

@testBlue @testBlue that selfie was extreme @asidj
@SouthwestAir no thanks! Thanks

Switching locations should not change sentiment

@testBlue I want you guys to be first to fly to @CubaCanada
@VirginAmerica I miss the Shearbird in @DenverDenver

Switching person names should not change sentiment

... Airport agents were horrendous. @united was our savior
@united 8602947, @united at http://t.co/5lbu/g000, thanks.

Source: @united, @united, @united, @united, and @united. "Airport Agents: Horrendous! Not Happy!"
with @united. "Airport Agents: Horrendous! Not Happy!"

Example: Robustness of sentiments

Evaluate if the model contains the expected invariants

Appending randomly generated handles or URLs should not affect sentiment

@testBlue that selfie was extreme @asidj
@united stuck because staff took a break? Not happy - https://t.co/vqjve

Small typos through swapping characters should not affect sentiment

@testBlue @testBlue that selfie was extreme @asidj
@SouthwestAir no thanks! Thanks

Switching locations should not change sentiment

@testBlue I want you guys to be first to fly to @CubaCanada
@VirginAmerica I miss the Shearbird in @DenverDenver

Switching person names should not change sentiment

... Airport agents were horrendous. @united was our savior
@united 8602947, @united at http://t.co/5lbu/g000, thanks.

Source: @united, @united, @united, @united, and @united. "Airport Agents: Horrendous! Not Happy!"
with @united. "Airport Agents: Horrendous! Not Happy!"

Domain-specific invariants

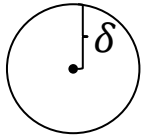
- Similar to what we considered earlier
- Credit rating should not be affected by gender
 - Applies to: $x \in data$
 - Morphism: $g(x) = \begin{cases} x[gender = male] & \text{if } x.gender = female \\ x[gender = female] & \text{if } x.gender = male \end{cases}$
 - Expected relationship: $f(g(x)) = f(x)$
- Less income should not lead to a better credit rating
 - Applies to: $x \in data, x.income \geq 1$
 - Morphism: $g(x) = x[income = x.income - 1]$
 - Expected relationship: $f(g(x)) \leq f(x)$
- ...

$x[y = z]$ denotes that the feature y of the instance x is set to z without changing any other feature
 $x.y$ denotes the value of feature y of the instance x


Robustness-related invariants

- Check if outcome fulfills some smoothness property with relation to training data
- Tiny modifications of training data should not change results
 - Applies to: $x \in data$
 - Morphism: $g(x) = x + \epsilon$ for a sufficiently small feature vector ϵ with $\|\epsilon\|_2 < \delta, \delta > 0, \delta \in \mathbb{R}$
 - Expected relationship: $f(g(x)) = f(x)$

Result does not change in a ball around each instance within the data


- Model output in region around training data within a certain range
 - Applies to: $x \in data$
 - Morphism: $g(x) = x + \epsilon$ for a feature vector ϵ with $\|\epsilon\|_2 < \delta, \delta > 0, \delta \in \mathbb{R}$
 - Expected relationship: $MIN \leq f(g(x)) \leq MAX$

No extreme values are observed for data sufficiently similar to data


- ...

Approaches for defining metamorphic tests

- Generate random test data
 - Usually easy
 - Should be generated to avoid error hiding
 - If the data is extreme and something fails, failure may be related to data (e.g., numeric issues), not property
 - Unclear if data or property the reason for the error
- Transform existing training, validation, and/or test data
- Adversarial learning
 - Gradient-based search for invariant violations
- Formal verification of invariants is an open research problem
 - Works, e.g., for simple constraints and small neural networks























Acting on invariant violations

- Usually based on weak correctness assumption
 - Some violations are usually unavoidable
- Sometimes strict requirements regarding violations
 - Safety
 - Fairness, robustness
- Representativeness of data relevant for deciding how to act
 - Random data may not be from the target distribution → violations may not be problematic
 - This requires very good knowledge (or strict assumption) about the target distribution!

More software testing concepts

Test coverage

JaCoCo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 org.jacoco.examples		58%		64%	24	53	97	193	19	38	6	12
 org.jacoco.core		97%		93%	112	1,410	116	3,376	20	715	2	138
 org.jacoco.agent.rt		76%		82%	32	122	69	319	21	74	7	20
 jacoco-maven-plugin		90%		80%	37	192	46	414	8	116	0	23
 org.jacoco.cli		97%		100%	4	109	10	275	4	74	0	20
 org.jacoco.report		99%		99%	4	572	2	1,345	1	371	0	64
 org.jacoco.ant		98%		99%	4	163	8	429	3	111	0	19
 org.jacoco.agent		86%		75%	2	10	3	27	0	6	0	1
Total	1,372 of 27,573	95%	153 of 2,187	93%	219	2,631	351	6,378	76	1,505	15	297

Measures how many lines/statements/decisions/... are executed (=covered) by the tests

Test effort driven by the structure of the source code!

Structural testing (aka white box testing, glass box testing)

- Typically aims to increase the coverage of tests
- Concept can also be applied to ML models
 - Does the test data cover all paths in a decision tree?
 - Are all neurons in a deep neural network activated at least once?
- Not always well defined
 - Linear regression?!
 - Support Vector Machine?!
 - Naive Bayes?!
- Automated generation of data to cover models possible
 - Data often not realistic → unclear how useful this is
 - Can be combined with adversarial learning to get more meaningful results

Regression testing

- For every bug, add a test case
- Increases likelihood that new bugs within the same code (so called regressions) are caught
- Execute regression tests after changes to find defects early
 - Ideally with continuous integration tools (e.g., nightly, for every commit, or for every pull request)
- Maps well to curating test data for ML
 - Especially useful for important populations and invariants!

Mutation analysis (aka mutation testing)

- Start with a working program (i.e., a program that passes its test suite)
- Automatically insert small modifications (mutation)
 - $a+b \rightarrow a-b$
 - $a < b \rightarrow a \leq b$
 - ...
- Check if test suite detects the modification
 - “Kills” the mutant
- Better test suites kill more mutants (mutation score)
- Early research exists
 - E.g., mutate model parameters (e.g., weights) or inputs
 - Often combined with coverage criteria
 - Practical usefulness and applications still unclear

Continuous Integration (CI)

The screenshot shows the GitHub Actions interface for the repository 'sherbold / data-science-crashkurs'. The workflow 'pages build and deployment' is shown, with a summary of jobs: 'build', 'report-build-status', and 'deploy'. The 'build' job is selected, showing its logs. The logs indicate that the job succeeded on 26 Apr in 7s. The workflow steps include 'Set up job' and 'Checkout'. The 'Set up job' step shows the current runner version, operating system, virtual environment, and the download of action repositories. The 'Checkout' step shows the repository being synced and the Git version being checked out.

Search or jump to... Pull requests Issues Marketplace Explore

sherbold / data-science-crashkurs Public

<> Code Issues Pull requests **Actions** Projects Wiki Security 238 Insights Settings

pages build and deployment pages build and deployment #3

Summary

Jobs

- build
- report-build-status
- deploy

build
succeeded on 26 Apr in 7s

Set up job

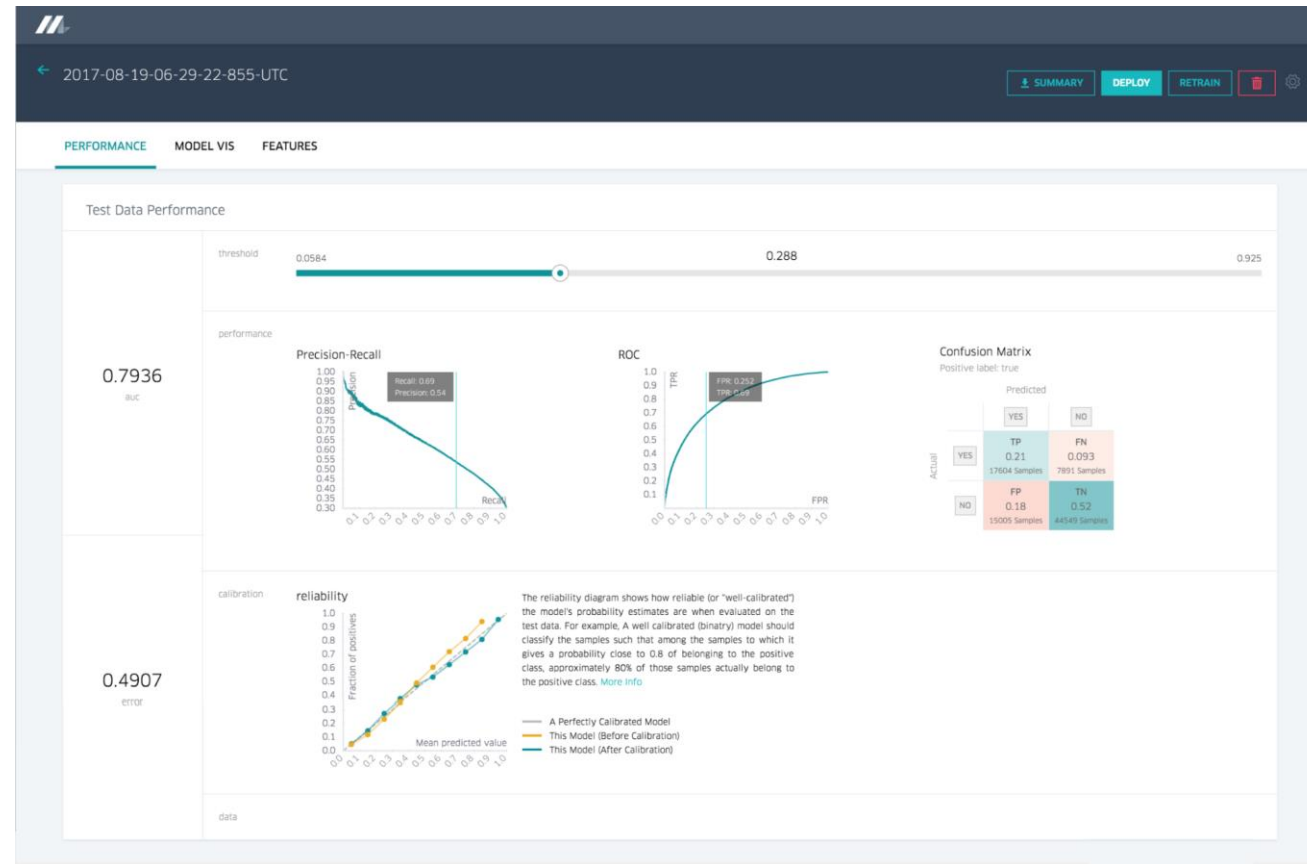
```
1 Current runner version: '2.290.1'
2 Operating System
6 Virtual Environment
11 Virtual Environment Provisioner
13 GITHUB_TOKEN Permissions
17 Secret source: Actions
18 Prepare workflow directory
19 Prepare all required actions
20 Getting action download info
21 Download action repository 'actions/checkout@v2' (SHA:7884fcad6b5d53d10323aee724dc68d8b9096a2e)
22 Download action repository 'actions/upload-artifact@main' (SHA:09a5d6a283da3e7c9f3253a5d4cdab2347712a66)
```

Checkout

```
1 Run actions/checkout@v2
13 Syncing repository: sherbold/data-science-crashkurs
14 Getting Git version info
18 Temporarily overriding HOME='/home/runner/work/_temp/fd123d80-b8f3-49bd-bd7f-4f5e8cc315d5' before making global git config changes
```


Automatically build and test software in case of changes

Works well to monitor changes of ML models



Michelangelo for the exploration of model performance for different builds

Dashboards for build and version comparison




[Github](#) [Docs](#)

Listing Price Prediction

Experiment ID: 0 Artifact Location: /Users/matei/mlflow/demo/mlruns/0

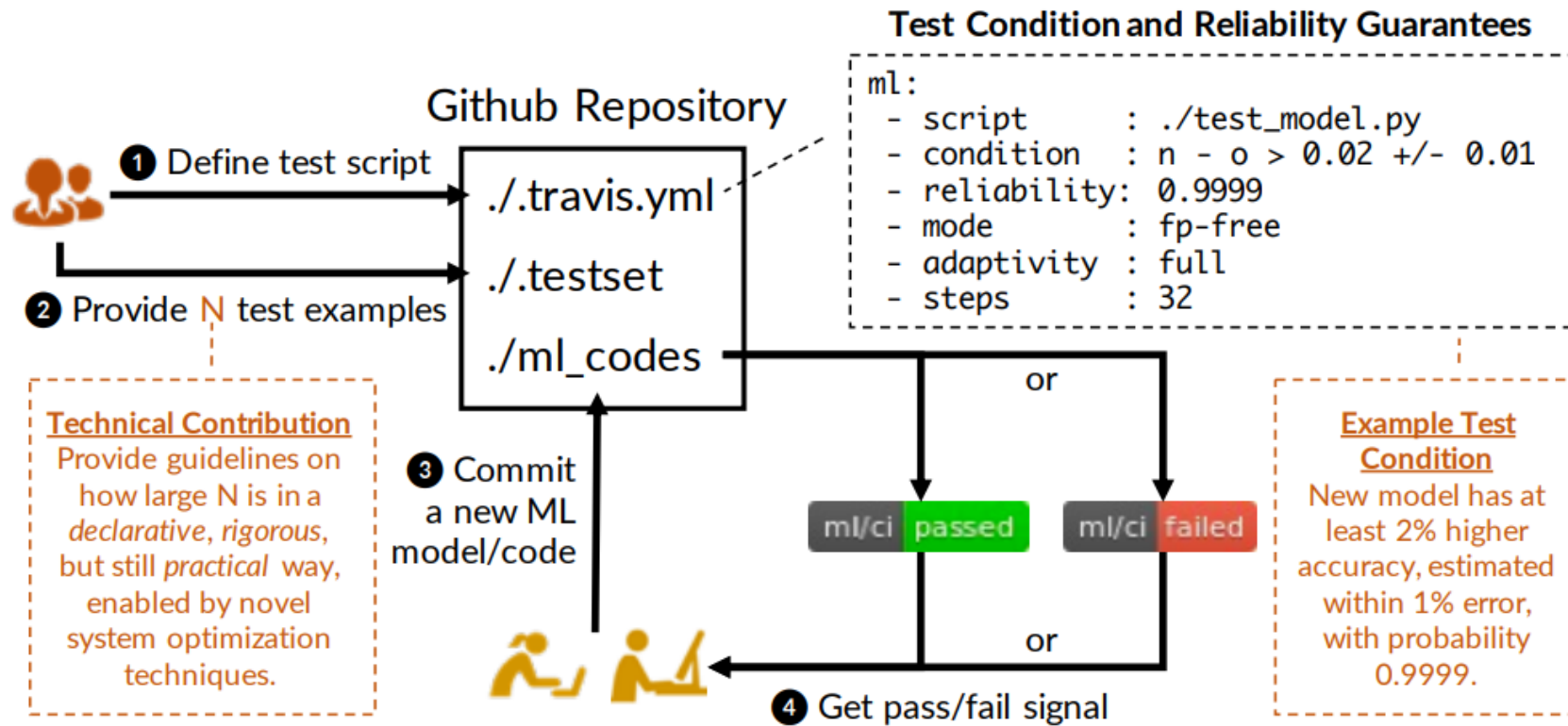
Search Runs: Search

Filter Params: Filter Metrics: Clear

4 matching runs Compare Selected Download CSV 

	Time	User	Source	Version	Parameters		Metrics		
					alpha	l1_ratio	MAE	R2	RMSE
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.2	84.27	0.277	158.1
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.2	0.5	84.08	0.264	159.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.5	84.12	0.272	158.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0	0	84.49	0.249	161.2

Specialized CI systems



Can automatically check for model regressions or if improvements are as expected
→ Special type of directly integrated test case!

Questions?

