
Practical Aspects of Data Science

Data Science Retreat - 2022/B29
Patrick Baier

Course Goal

Prepare you to be ready for the daily work in a data science job

→ **Model Learning (Day 1)**

Model training, Evaluation metrics, imbalanced data problems

→ **Model Operation (Day 2)**

Probability calibration, Model deployment, missing features, monitoring

Schedule - Day 2

09:30 - 13:30 Probability Calibration (including 1 hour lunch break)

13:30 - 15:00 Model Deployment

15:00 - 16:30 Data Imputation

16:30 - 18:00 Monitoring + DS Organization

Probability Calibration

Problem description

In binary classification (class *zero* and class *one*), a model gives us the probability that a data point belongs to class one (i.e. $p = 0.7$). This probability gives us some kind of confidence on the prediction.

Question: How realistic is this probability?

Problem:

Not all classifiers provide well-calibrated probabilities, some being over-confident while others being under-confident. Thus, a separate calibration of predicted probabilities is often desirable as a postprocessing step.

Calibration

Why is this important?

Many real-world problems require a realistic probability rather than a binary decision.

- Estimation of money at risk (= $p_{\text{return}} \cdot \text{basketValue}$)
- Estimate click probability of a banner

Different machine learning models are better or less well calibrated:

- Logistic regression: Naturally gives well-calibrated predictions.
- Tree ensembles: Not so good calibrated predictions.
- Oversampling leads to unrealistic probabilities.

Calibration

How can we check if a model is well calibrated?

1. The **Brier Score** evaluates the quality of probabilistic predictions.
Output: A number
2. The **Calibration Plot** shows in which areas the model is well calibrated.
Output: A plot.

Brier Score

Example:

- We have two models that correctly predicted the return of one order.
- The first model outputs probability 0.51
- The second model outputs probability 0.93.
- They are both correct (assuming 0.5 threshold).

The second model does a better job, since its probability is closer to the true outcome (=1). We can measure this using the **Brier Score**.

Brier Score

The Brier score measures the mean squared difference between the predicted probability and the actual outcome:

$$BS = \frac{1}{N} \sum_{t=1}^N (f_t - o_t)^2$$

f_t	probability (output from model)
o_t	observed probability
N	number of data points

The lower the Brier score, the better a model is calibrated.

Brier Score

The Brier score measures the mean squared difference between the predicted probability and the actual outcome:

$$BS = \frac{1}{N} \sum_{t=1}^N (f_t - o_t)^2$$

f_t	probability (output from model)
o_t	observed probability
N	number of data points

What is this this?

Answer: The label of the data point, which is 0 or 1.

Calibration Plot

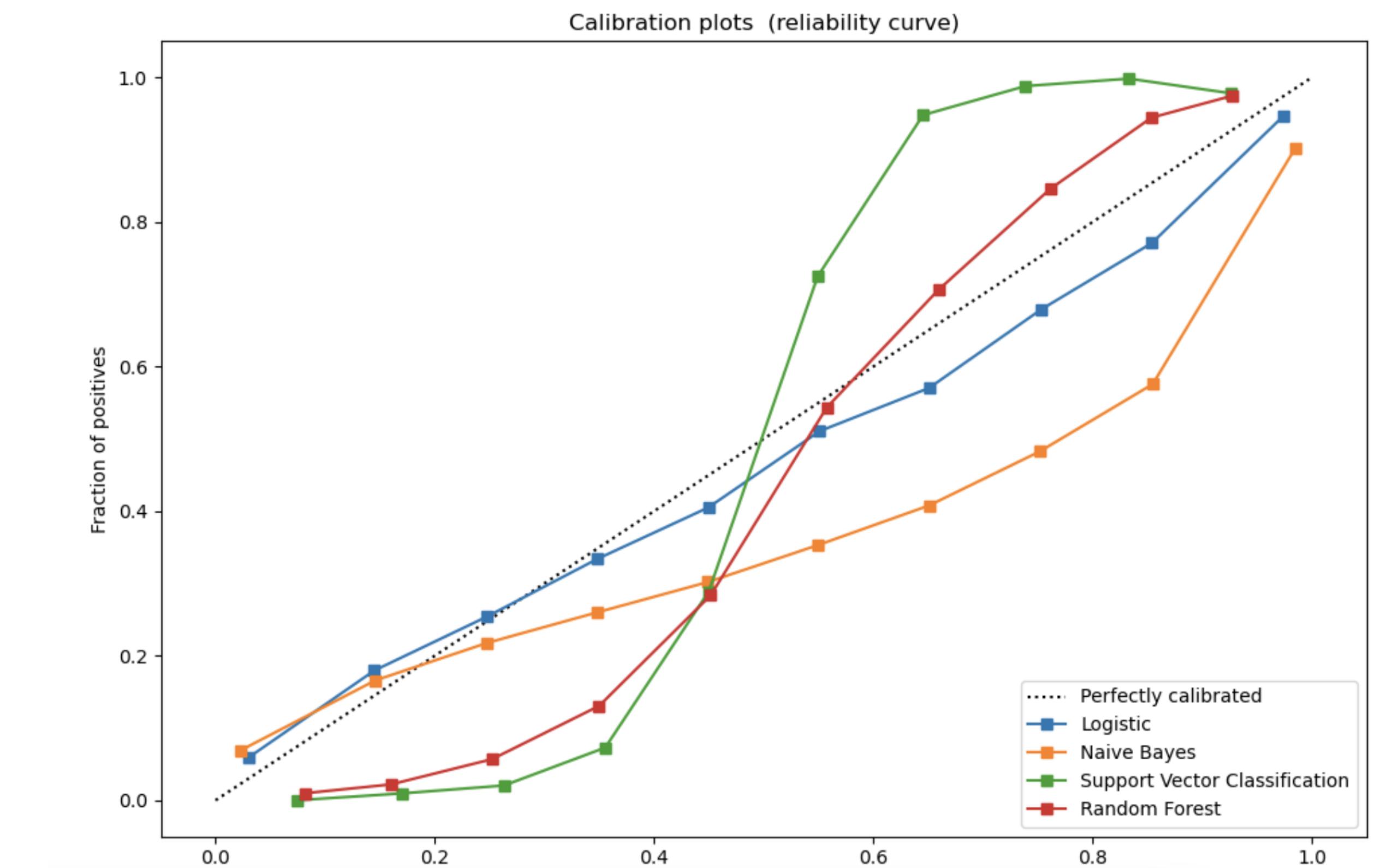
The calibration plot shows the relation between:

- Predicted probability (x-axis)
- Real probability (y-axis)

How to construct this?

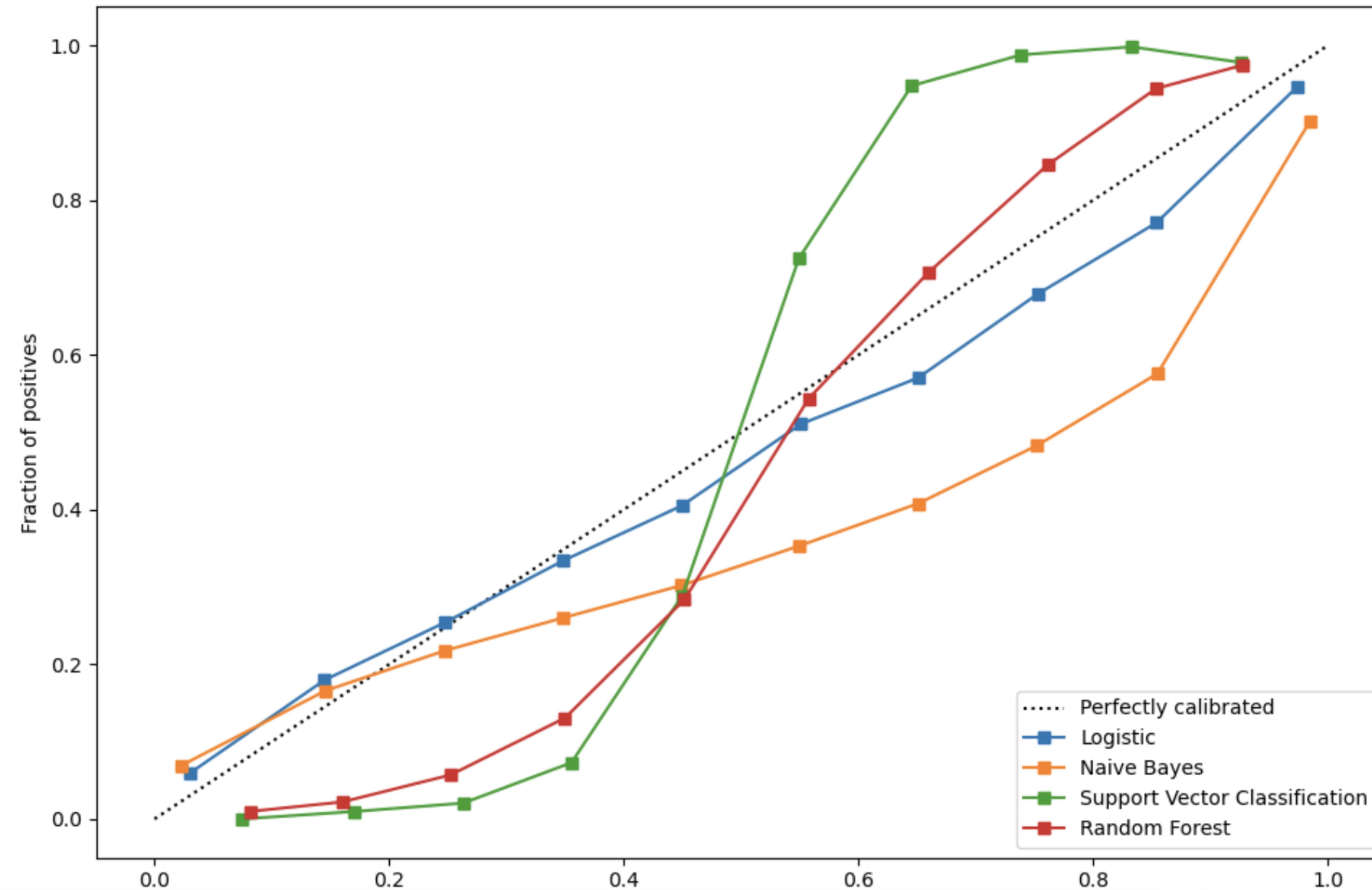
Problem: We do not have a real probability („fractions of positives“ in the plot), just 1 or 0.

Answer: Use buckets to group similar orders together.



See zoom in on next slide

Calibration plots (reliability curve)



prediction	real label
0.9	1
0.8	1
0.85	0
0.7	1
0.69	1
...	...
...	...
...	...
...	...
...	...
...	...
...	...
...	...
0.05	0

bucketSize = 5

- Sort data points by prediction probability.
- Group consecutive data points together into buckets.
- For every bucket calculate:

Bucket 1:

$$p_{\text{real}} = 4/5 = 0.8$$

$$p_{\text{pred}} = (0.9 + 0.8 + 0.85 + 0.7 + 0.69)/5 = 0.78$$

prediction	real label
0.9	1
0.8	1
0.85	0
0.7	1
0.69	1
...	...
...	...
...	...
...	...
...	...
...	...
0.05	0

bucketSize = 5

Bucket 1:

$$\begin{aligned} p_{\text{real}} &= 0.8 && \Rightarrow y_1 \\ p_{\text{pred}} &= 0.78 && \Rightarrow x_1 \end{aligned}$$

Bucket 2:

$$\begin{aligned} p_{\text{real}} &= 0.5 && \Rightarrow y_2 \\ p_{\text{pred}} &= 0.3 && \Rightarrow x_2 \end{aligned}$$

Bucket 3:

$$\begin{aligned} p_{\text{real}} &= 0.1 && \Rightarrow y_3 \\ p_{\text{pred}} &= 0.2 && \Rightarrow x_3 \end{aligned}$$

=> Three points in the calibration plot.

Calibration Plot

Summary of steps:

1. Predict on test data.
2. Sort by predicted probabilities in descending order.
3. Create buckets for every x (= hyperparameter) data points.
4. Within every bucket calculate:
 - a. p_{real} : real probability ($\# \text{ones} / \# \text{all}$)
 - b. p_{pred} : average predicted probability
5. Use the p_{real} as y values, use the p_{pred} as x values in a plot.

How to get good calibration?

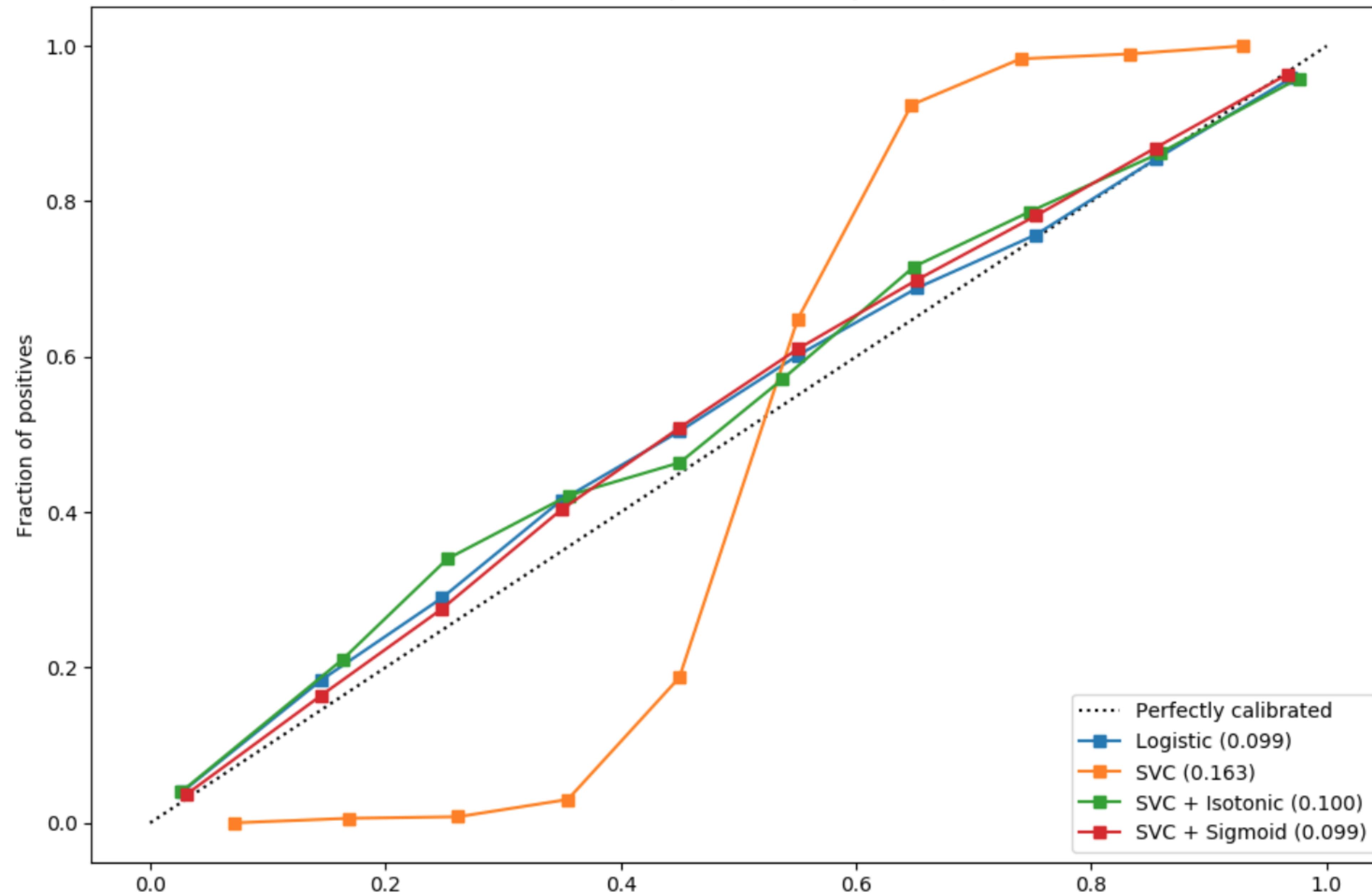
Learn calibration function on top of existing ML model on the training data.

X: Output probability of the model on training data
y: Real label

There are different approaches to this:

1. Sigmoid calibration: Use Logistic Regression to learn mapping X to y.
2. Isotonic calibration: Use [Isotonic Regression](#) to learn mapping X to y.

Calibration plots (reliability curve)

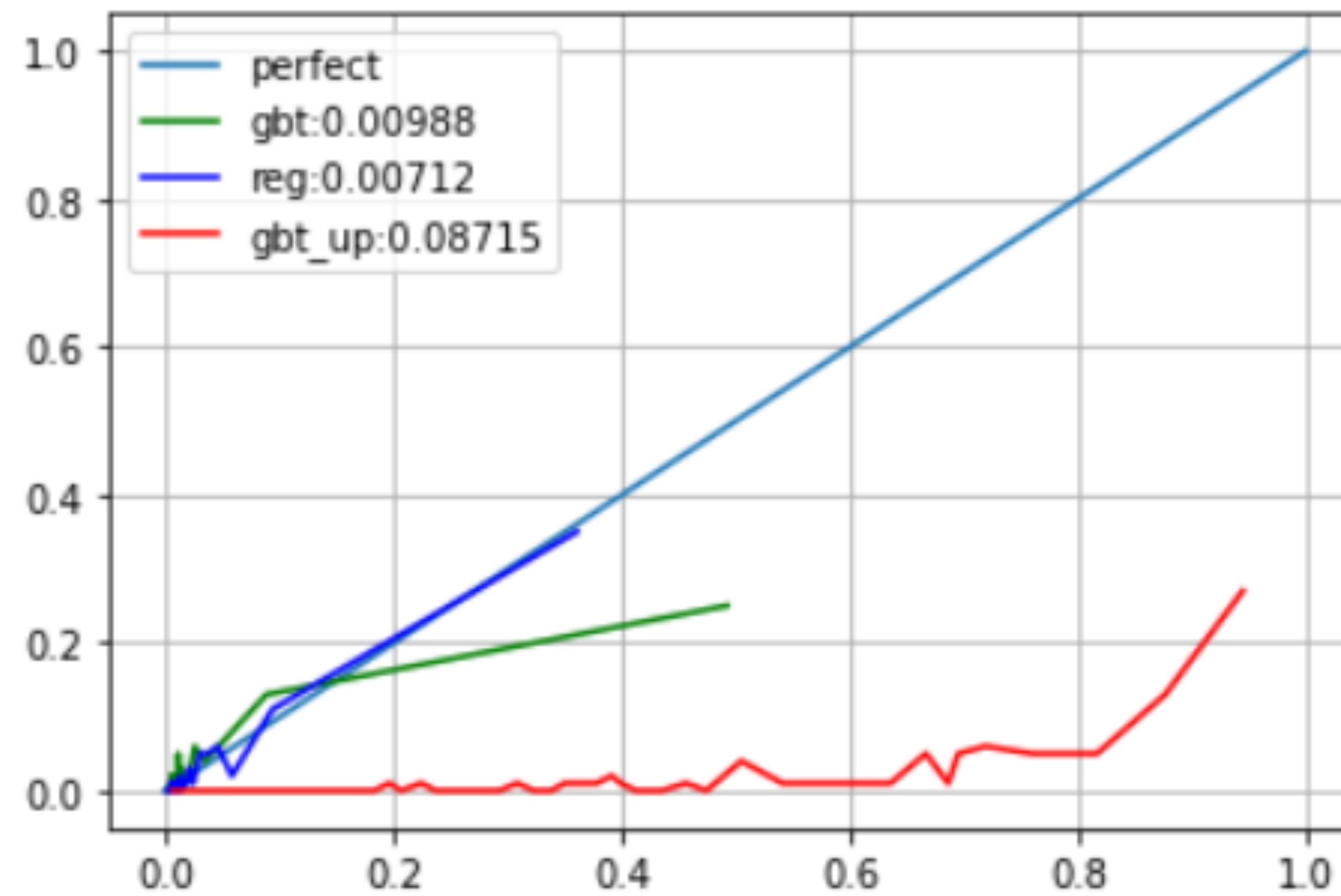


Task 4

1. Calculate the Brier score for the logistic regression and GBT from task 1.
Also do this for the upsampled GBT from task 3.
2. Plot the calibration plot for those three models. Use buckets of 100 consecutive predictions. (see next slide for the expected outcome)
3. Calibrate the upsampled GBT using the class
[CalibratedClassifierCV](#) from sklearn.
Calculate the Brier score on the test data and the calibration plot.

Task 4

Real probability
(fraction of
positives per
bucket)



Predicted probability
(Avg. prediction per bucket)

Task 4

4. How does this classifier behave in terms of:
 - Discriminative power? (area under the roc)
 - Calibration?
5. What would be an classifier that has the opposite characteristics?

p prediction	real label
0.3	1
0.2	1
0.10	1
0.09	1
0.06	0
0.05	0
0.04	0
0.03	0
0.02	0
0.01	0

Calibration Resources

More resources about probability calibration:

- <https://www.svds.com/classifiers2/>
- <http://scikit-learn.org/stable/modules/calibration.html>
- <https://machinelearningmastery.com/calibrated-classification-model-in-scikit-learn/>
- <http://tech.magnetic.com/2015/06/click-prediction-with-vowpal-wabbit.html>

Model Deployment

Real-time Predictions

Remember our requirement:

Built a binary classification model that predicts **in real-time** the probability if a customer returns the order.

- Currently our model only lives in the Jupyter notebook.
 - How do we make our model accessible to the world?
- Solution: We build a micro-service that contains our model.
- Input: Feature vector
 - Output: Return probability

Webservice

A web service (or micro-service) is a program that receives requests and answers them.

Most famous: web server, i.e. receive requests for HTML pages (from a browser), sends back HTML pages.



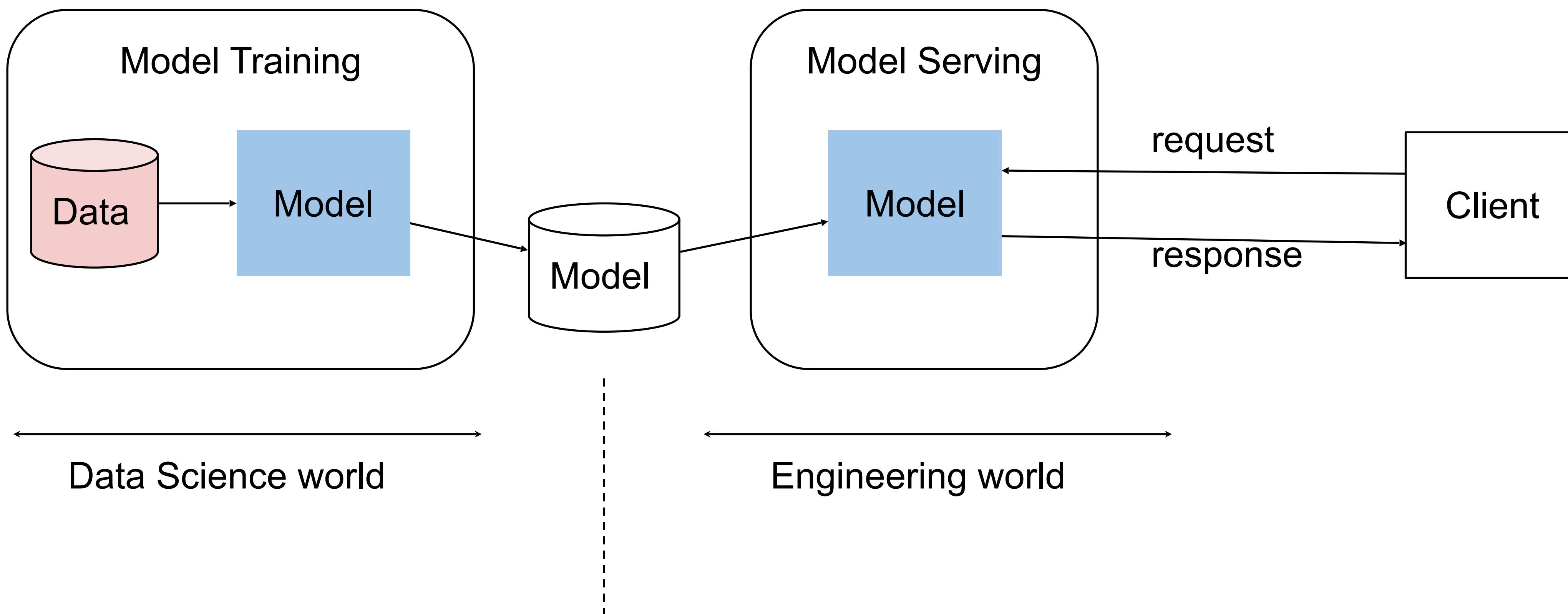
Weservice

We provide our model to the outside world through such a service such that other people can send a request in the form:

192.168.1.204:5000/predict
 / / ↑
IP address Port of service Endpoint for predictions

For this, we have to encapsulate our prediction model into a such a service.

Model in Production



Flask Webservice



Flask
web development,
one drop at a time

Let's build a REST service for predictions:

Bind
method to
predict
endpoint of
server

```
@app.route('/predict', methods=['POST'])
def predict():
    basket = request.json['basket']
    zipCode = request.json['zipCode']
    totalAmount = request.json['totalAmount']
    p = probability(basket, zipCode, totalAmount)
    return jsonify({'probability': p}), 201
```

Return return probability to client (as json)

Extract
fields from
request
that was
sent by
client

Test Request

Send a request to the server (bash script *request.test*):

```
curl -i -H "Content-Type: application/json" -X POST -d '{"transactionId":  
6304965406, "basket": [4, 3, 0, 3, 1, 1, 2, 0, 2], "zipCode": 2729,  
"totalAmount": 12}' http://localhost:5000/predict
```

target address

request data

Flask will make the request data available in Python as request object.

Task 1

1. Get the server up and running on your machine.

Execute in a shell: `python server.py`

1. Execute `request.test` to send a requests to your server:

Execute in a different shell: `source request.test`

For Windows users:

```
> python server.py  
> curl -i -H "Content-Type: application/json" -X POST -d "{\"transactionId\": 6304965406, \"basket\": [4, 3, 0, 3, 1, 1, 2, 0, 2], \"zipCode\": 2729, \"totalAmount\": 12}" http://localhost:5000/predict
```

Task 2

1. Extend the server to handle requests using the ML model.
 - Save your trained model in the jupyter notebook using joblib:
http://scikit-learn.org/stable/modules/model_persistence.html
 - Load the model into your server.
 - Predict the probability with the model on the incoming request.
2. Run the request simulation tool to test your server:
 - Execute in a shell: `python3 loadSimulator.py`
 - Why is it failing for some requests?
3. Bonus: Return a calibrated probability.

Missing Features

Missing Features

We often have data points where one or more features are missing:

$\langle x_1, x_2, \text{null}, x_4 \rangle$

This can happen at:

1. Training time

→ New feature (next slide), noisy data, optional fields, join operations, ...

2. (Live) Prediction time

→ Client cannot provide all features. Possible reasons:

Network timeouts, database timeouts, optional fields, ...

What can we do?

Feature is missing in training:

- Drop data point
- Value imputation
- Model imputation
- Missing indication

Feature is missing in live system:

- Live value Imputation
- Multiple models

Drop Data Points

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1

- In the live system that means that we do not predict on the data point (usually not the best solution!).
- In the training data it can drastically reduce the training data size or even introduce data biases!

Training: Value Imputation

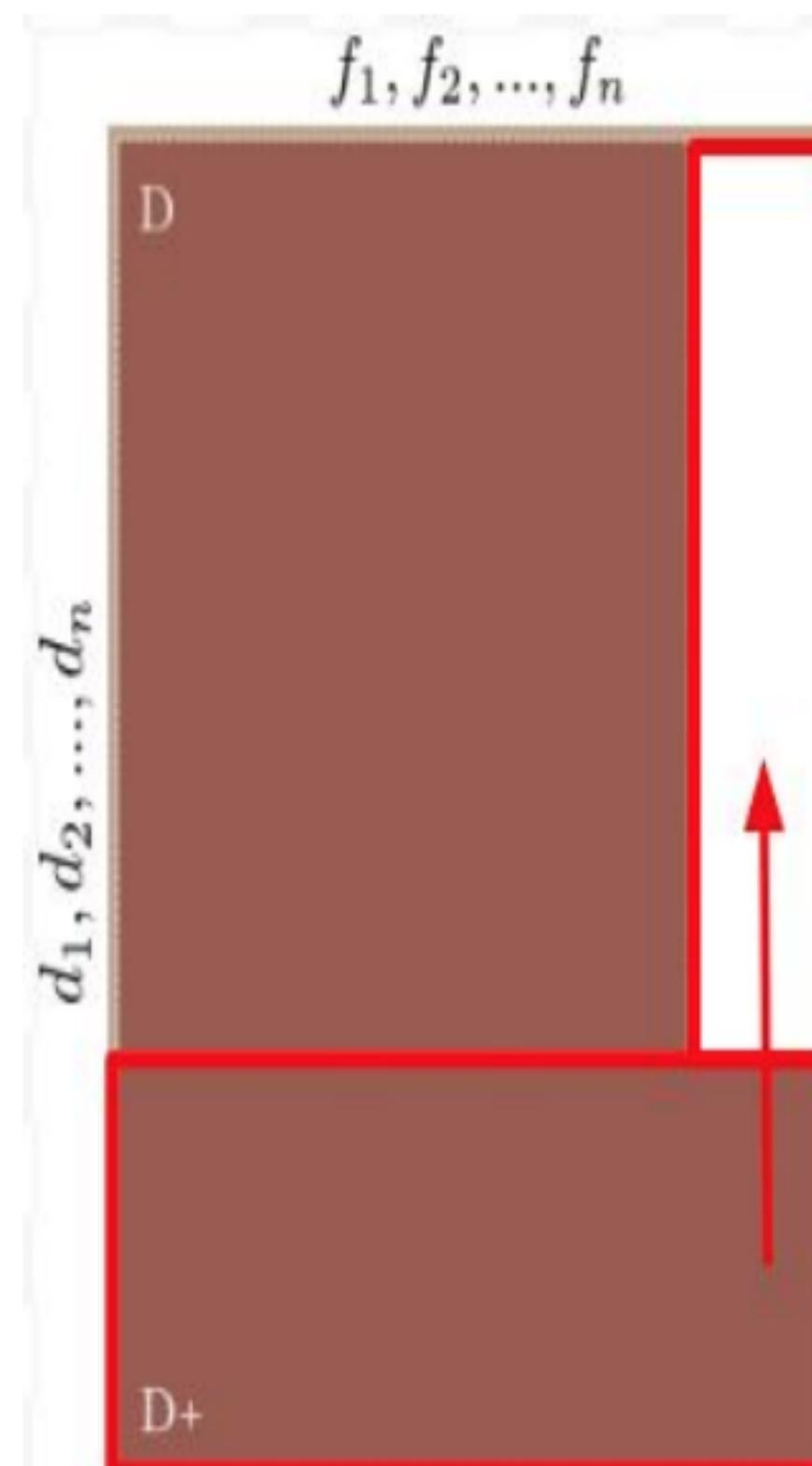
Fill up a feature that is not available with:

- median values of training data
 - mean value of training data
- }
- Continuous feature
-
- median values of training data
(if feature is ordinal)
 - majority value of training data
- }
- Discrete feature

Training: Value Imputation

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	116.7	33.6	0.627	50	1
1	1	85.0	66.0	29.0	116.7	26.6	0.351	31	0
2	8	183.0	64.0	30.8	116.7	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	30.8	116.7	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1

Training: Model Imputation



Model imputation:

- train model on complete data points
- target variable is the missing feature
- predict missing values for incomplete data points

Live: Multiple models

Learn models with reduced features:

Normal model:

$$f(x_1, x_2, x_3, x_4)$$

Reduced models:

$$f(x_2, x_3, x_4), f(x_1, x_3, x_4), \dots$$

$$f(x_3, x_4), f(x_2, x_4),$$

$$f(x_2, x_3), \dots$$

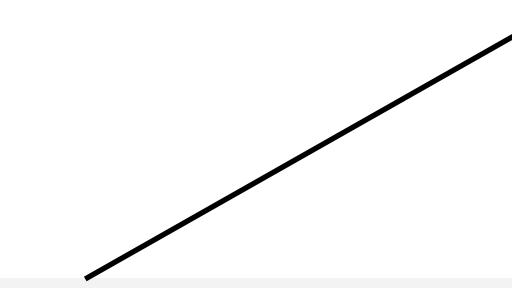
$$f(x_1), f(x_2), \dots$$

Choose at runtime maximal model with available feature set

- Huge overhead (2^n different models), but: best performance.
- Tradeoff: Only learn reduced models for often failing features.

Live: Value imputation

Data point with missing feature: $f(x_1, \text{null}, x_3, x_4)$



Replace the missing value with the mean/majority value for that feature that was observed in the training data.

- You need to have the imputation values in your server code.
- Imputation values need to be stored during training and loaded into the live model.

Task

1. Extend the web server to be able to deal with missing features:
 - Create an value imputer that is built on top of the training data.
 - Use this imputer in the flask server.
2. Run the request simulation tool again.
3. Bonus: Implement a /metrics endpoint in Flask that returns how often each feature got imputed in the last 100 requests.*

Note: Do not hard-code the imputation values in the flask server (use an dict-object that is loaded on startup).

* If you use “GET” instead of “POST” you can access the metrics endpoint from your browser on: <http://localhost:5000/metrics>

Monitoring

Motivation

Let's deploy a model for return prediction in an online shop!

Steps along the way:

1. Collect training data.
2. Train a model.
3. Deploy it to production.

Motivation

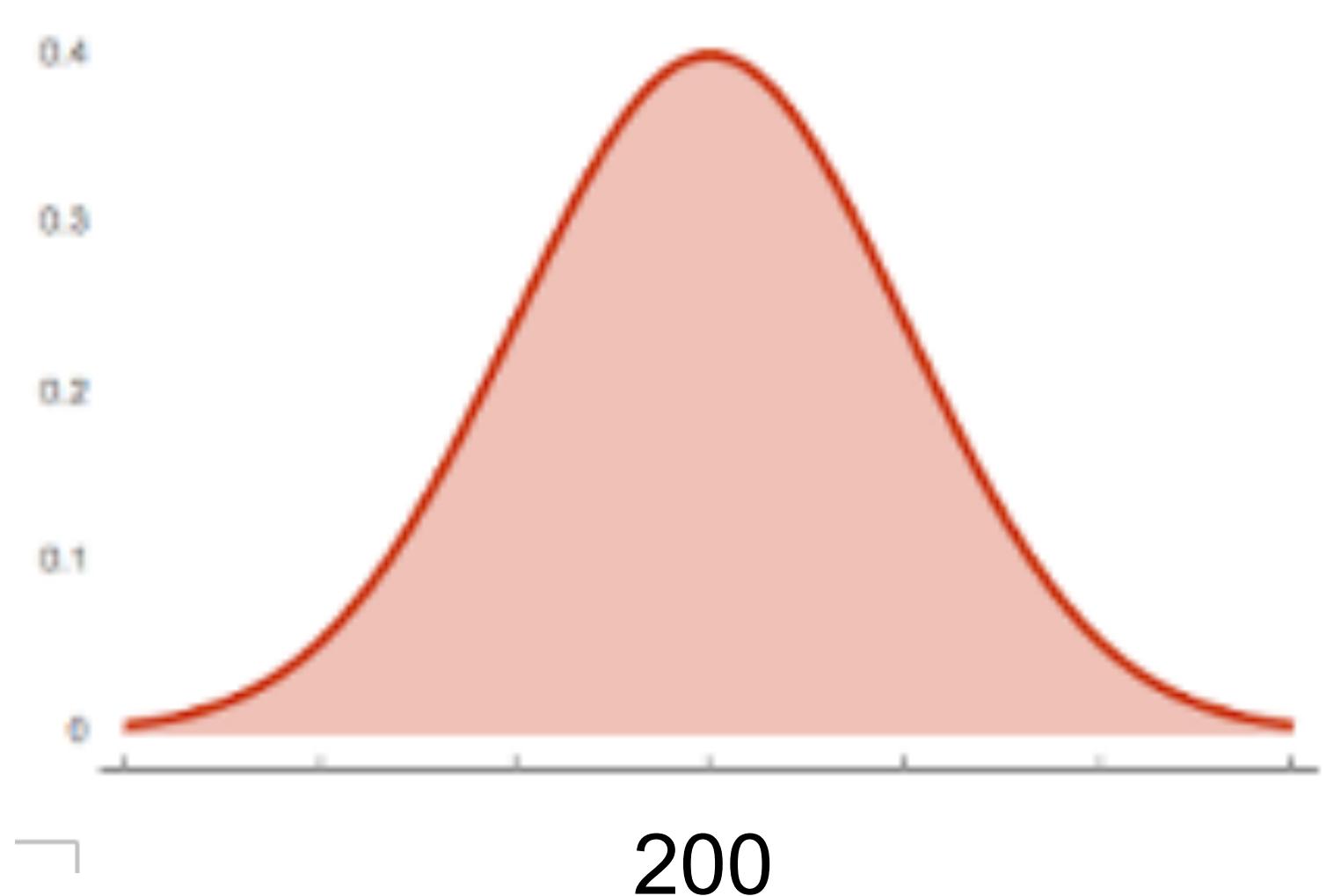
Our training data:

#	Feature-1	Time-to-order [s]	...	Feature-N	Label
1	2	300	...	1	no return
2	1	5	...	0	return
3	3	120	...	0	no return
4	2	200	...	1	no return
5	1	250	...	0	return
...

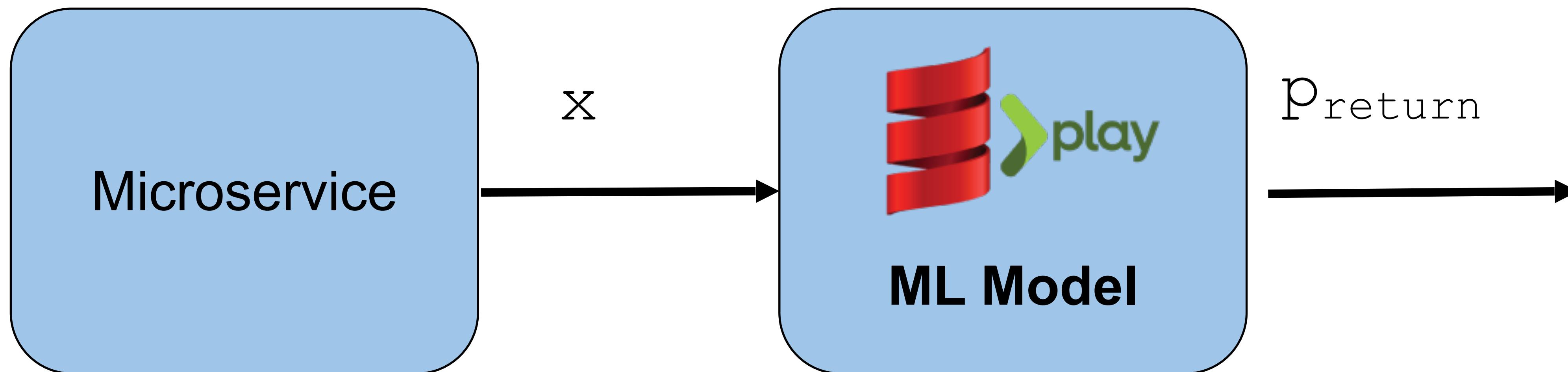
Motivation

Time-to-order [s]
300
5
120
200
250
...

Distribution of feature in training data



Motivation



Once we are live, we get features x sent over by a different microservice in real-time.



✓ CPU usage
✓ Memory usage
✓ Latency
....

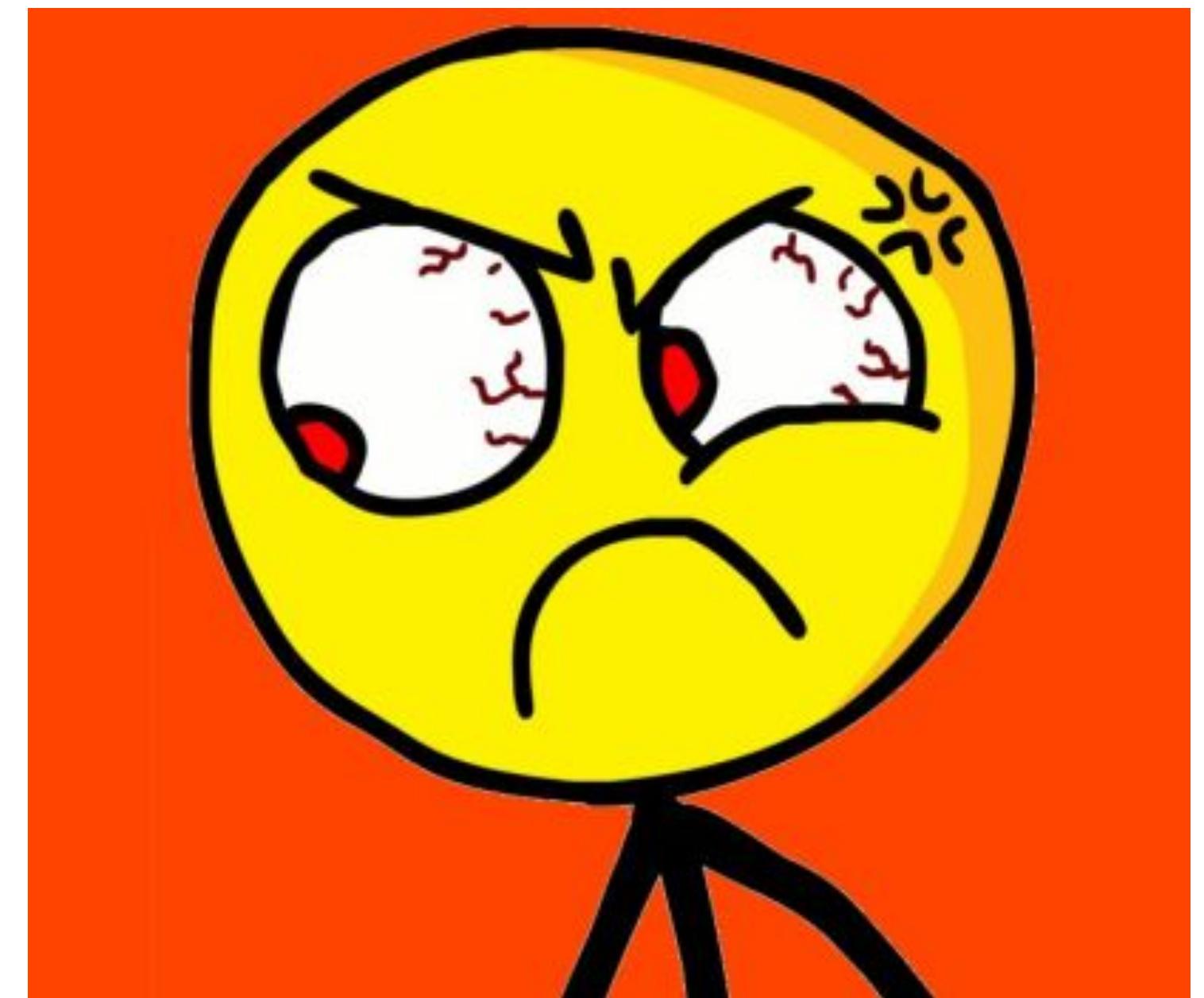
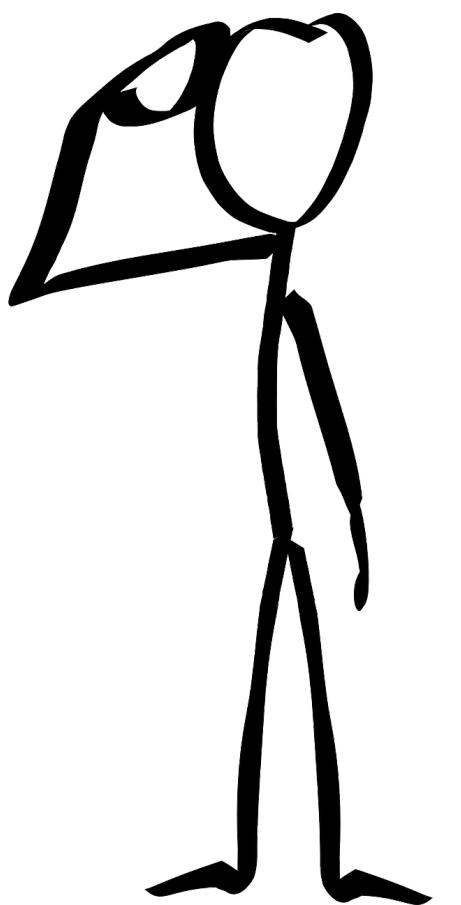


Motivation

Some weeks later, people are angry:

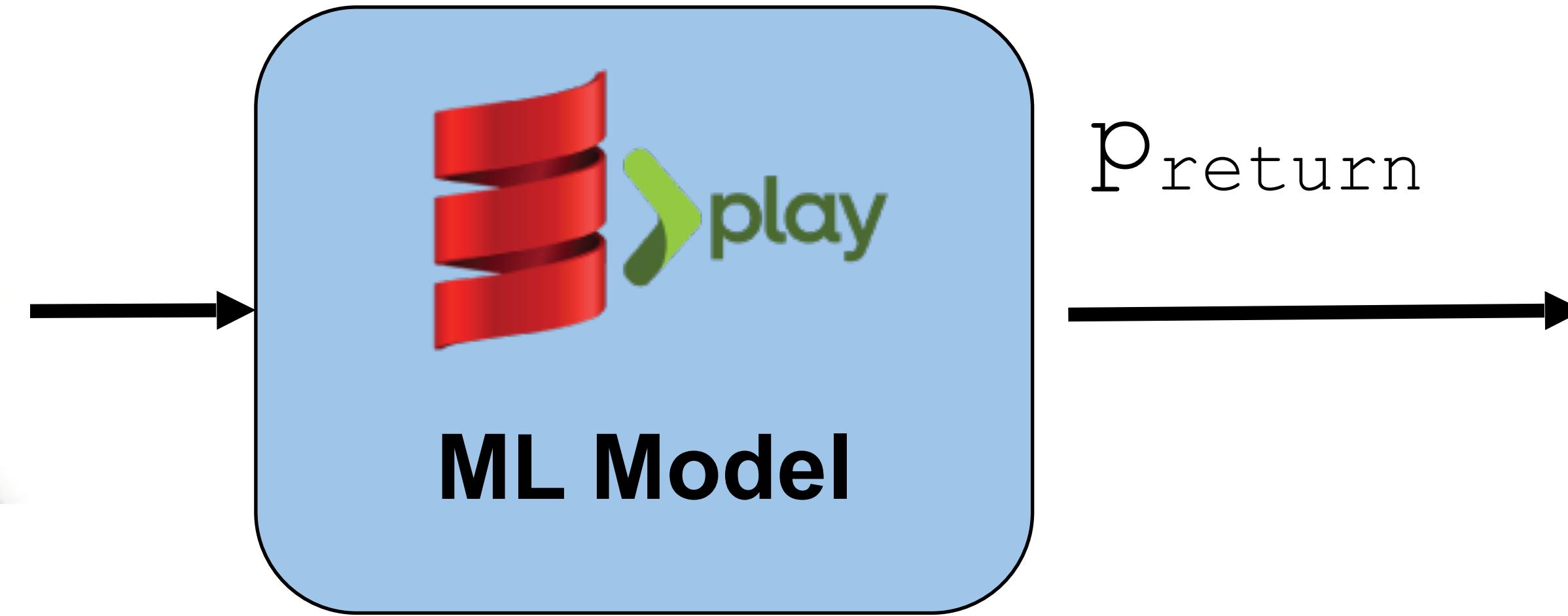
“We fail to detect returns, our business is ruined!”

What happened?



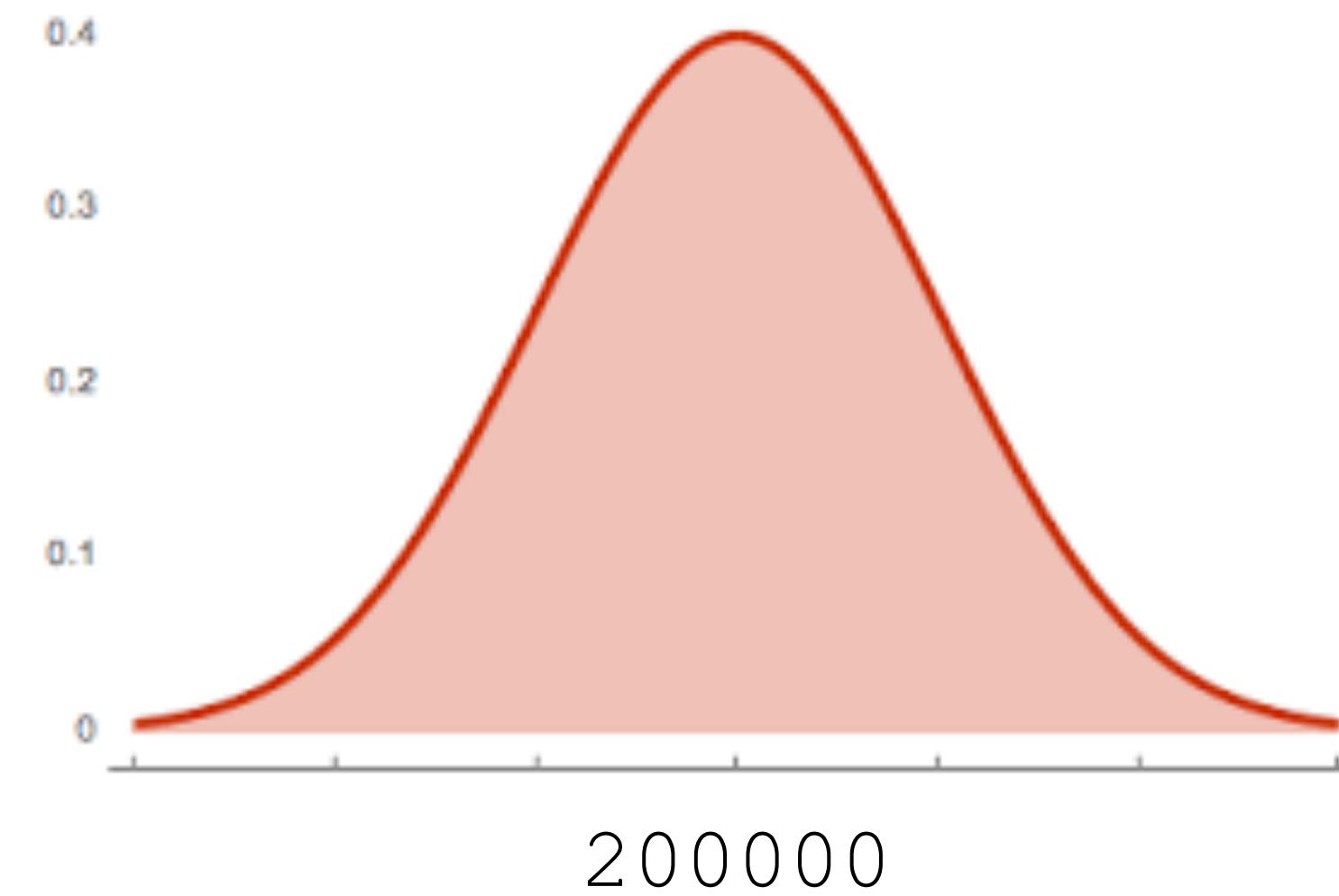


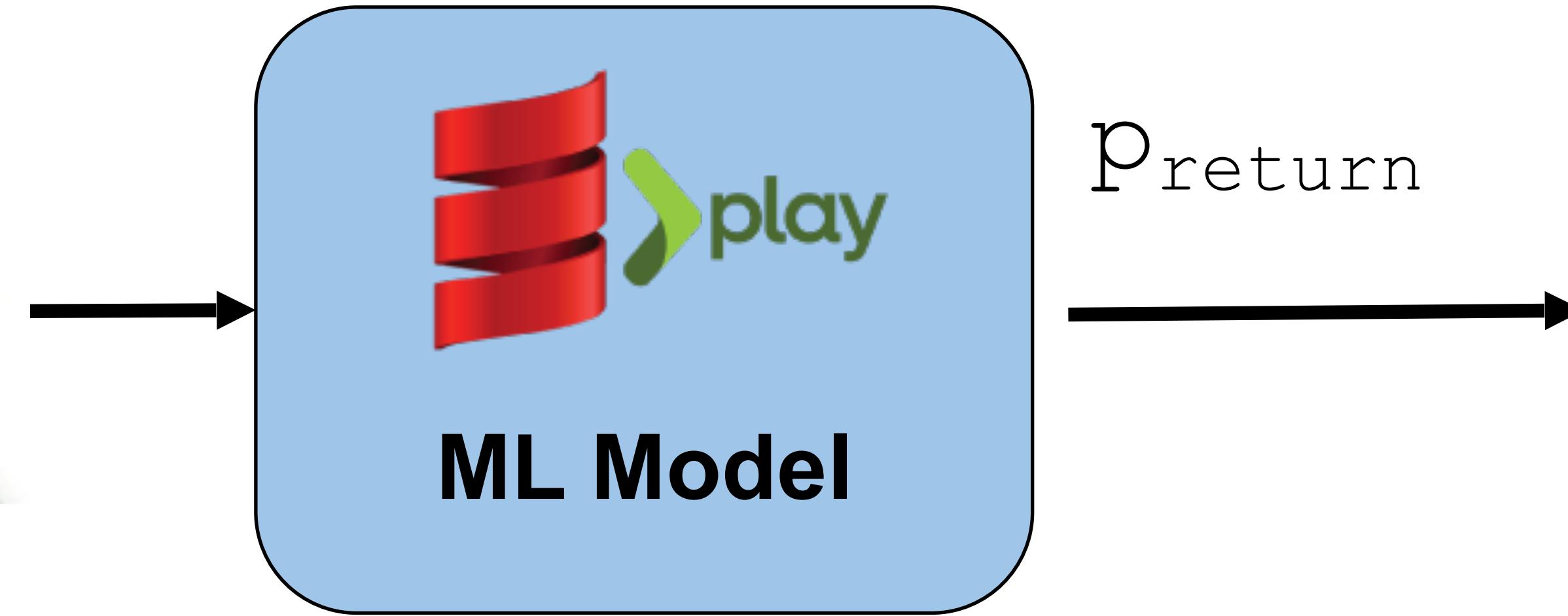
✓ CPU usage
✓ Memory usage
✓ Latency
....



Preturn

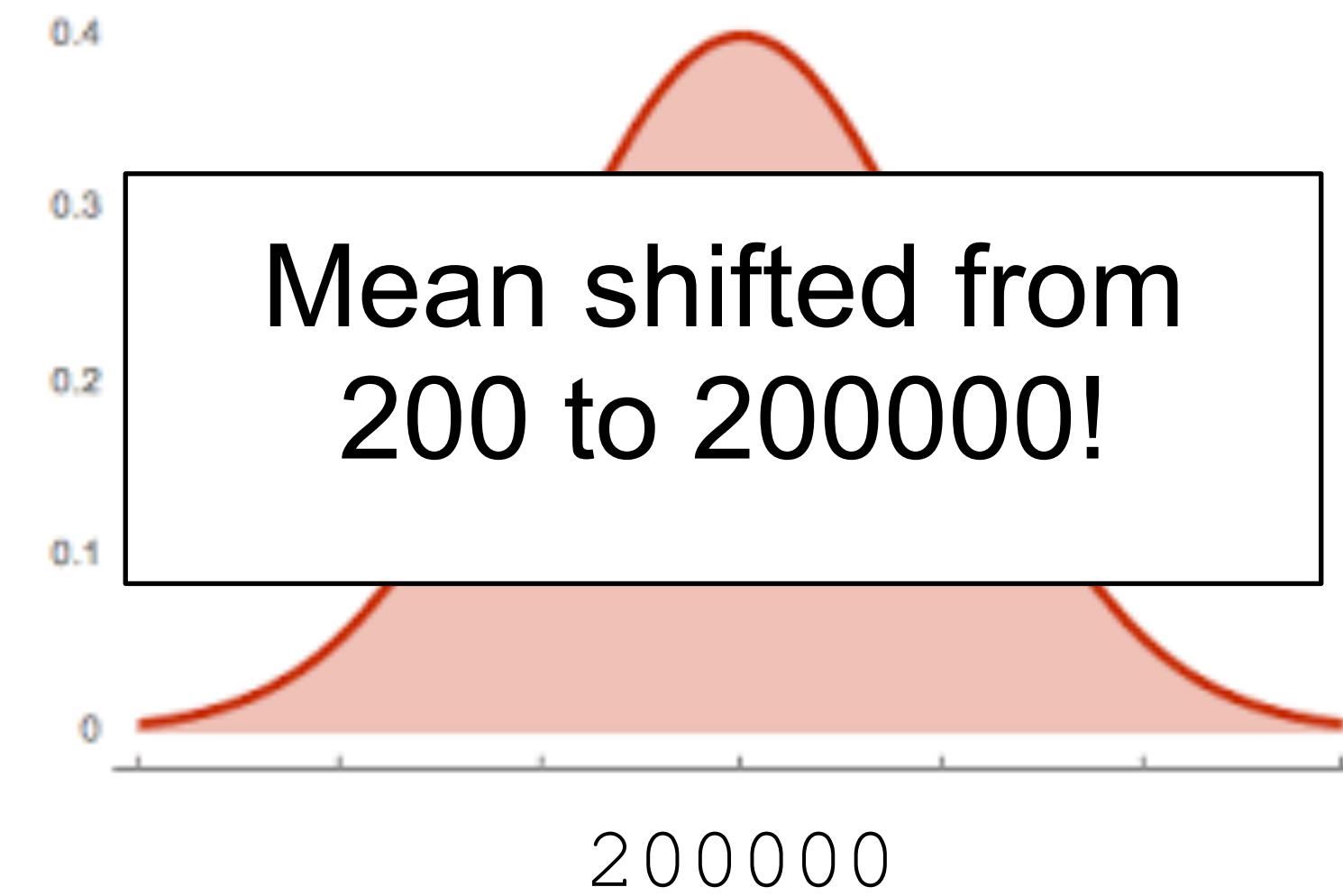
Time-to-order
300000
5000
120000
...

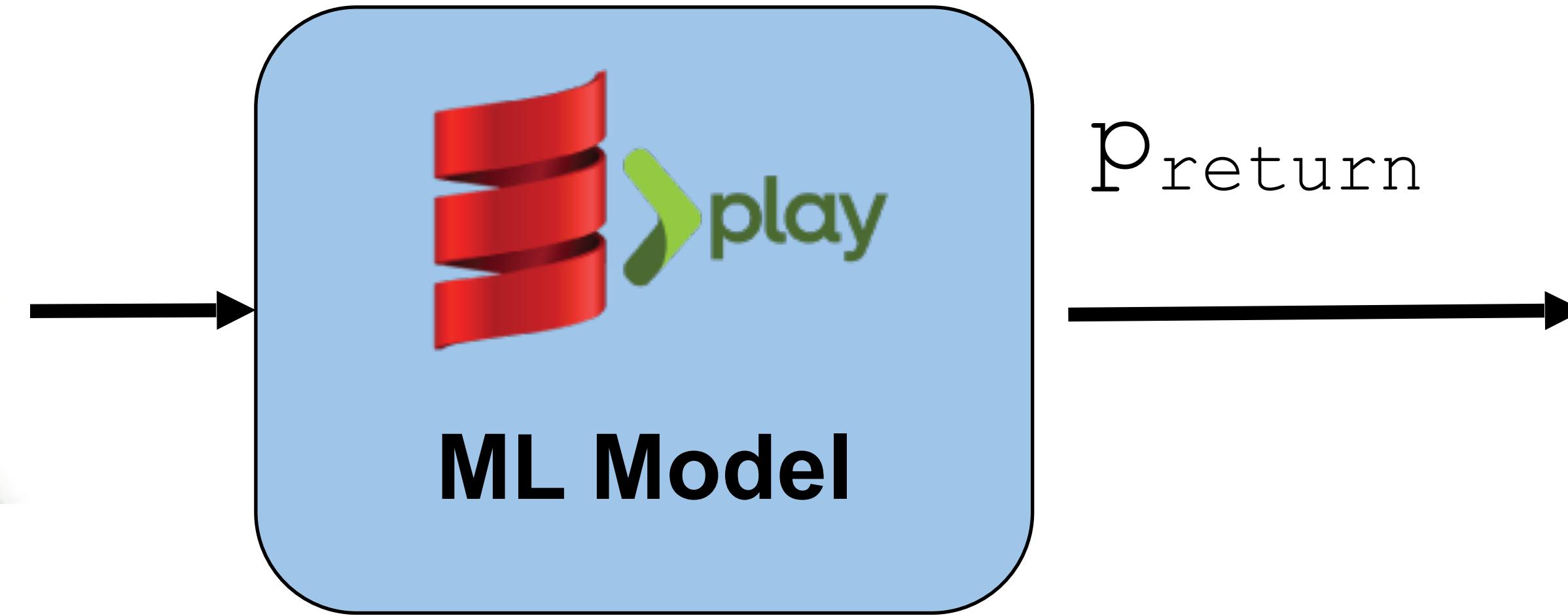




Preturn

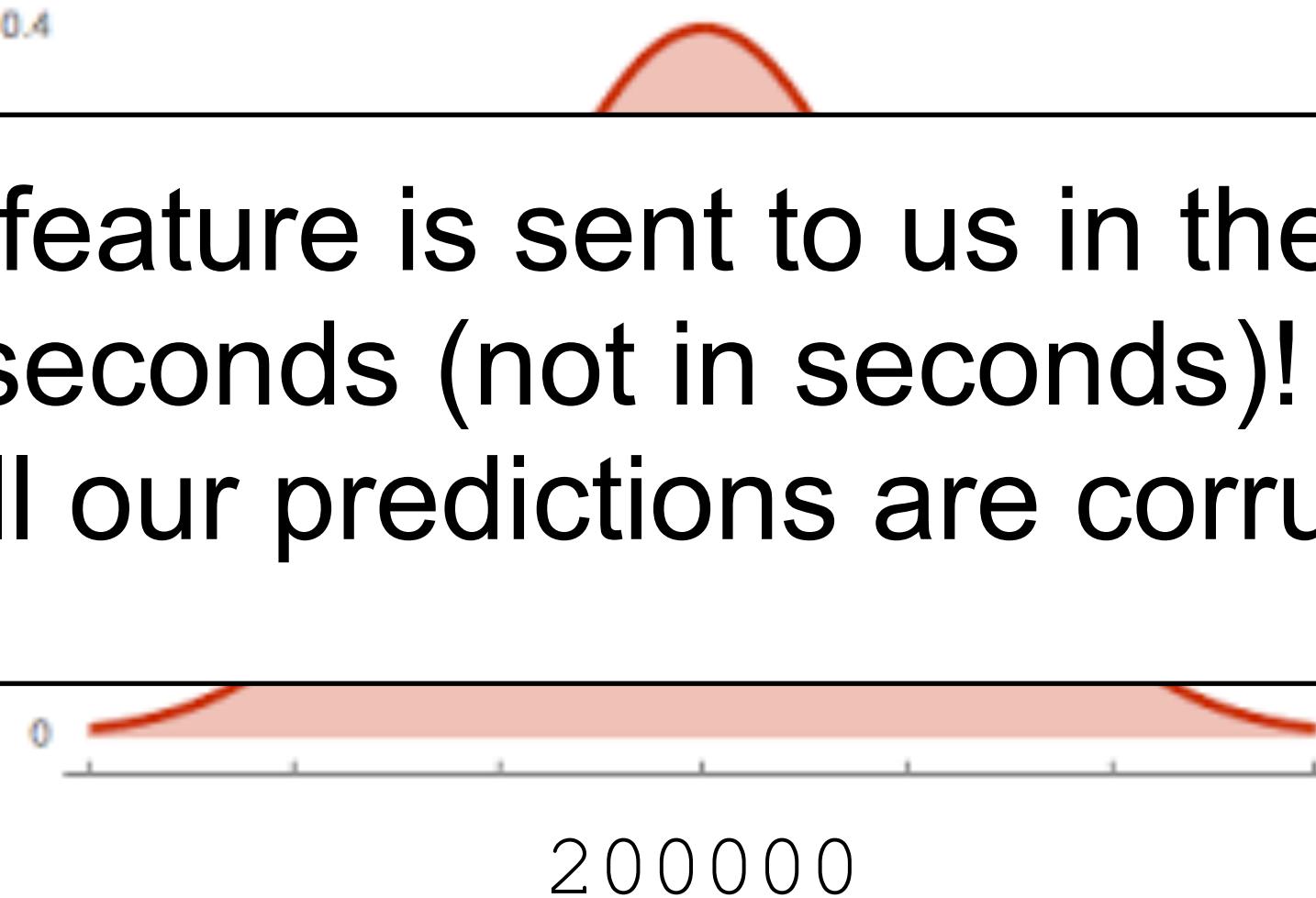
Time-to-order
300000
5000
120000
...





Time-to-order
300000
5000
120000
...

The feature is sent to us in the unit of milliseconds (not in seconds)!
→ All our predictions are corrupt!



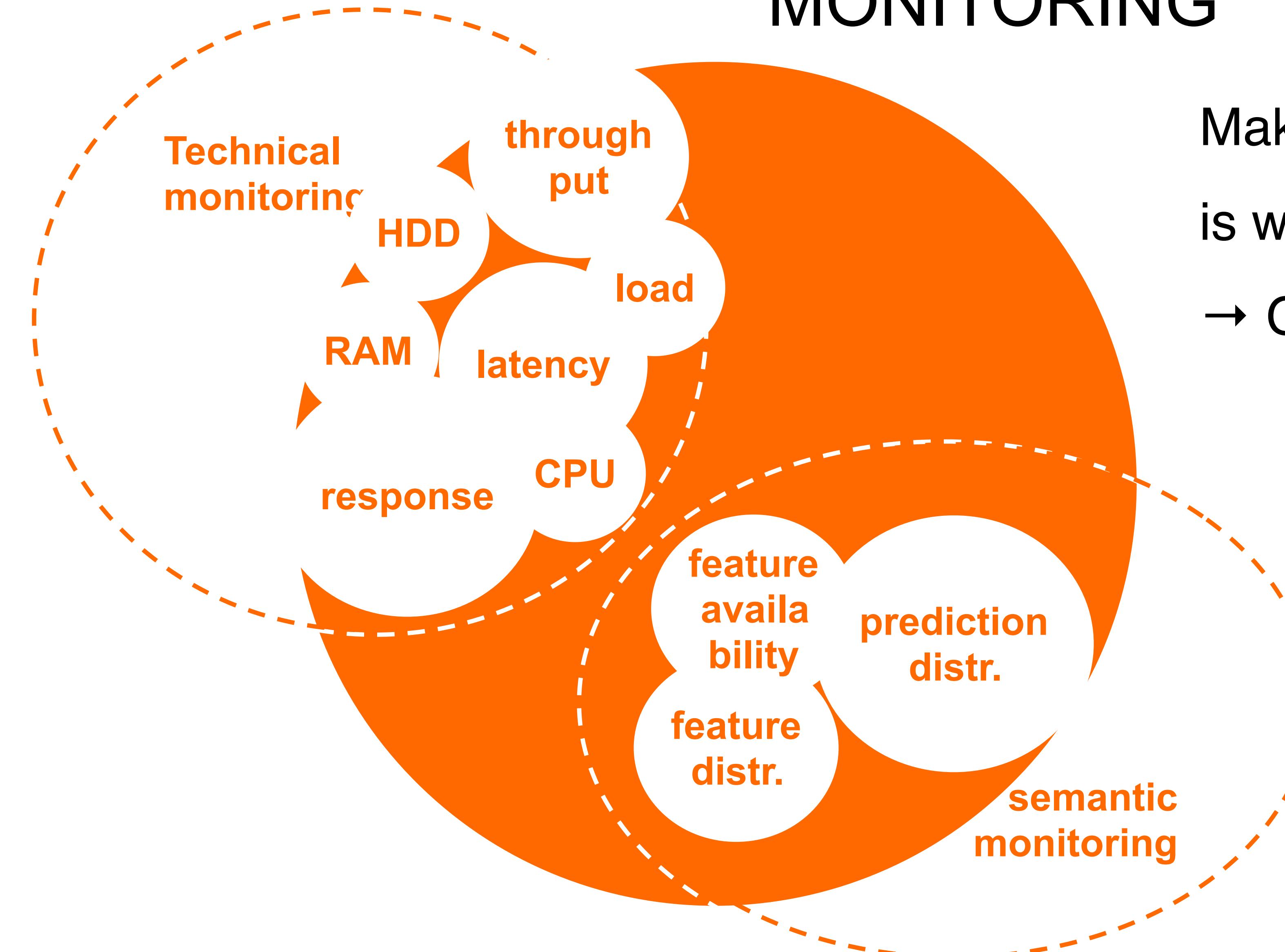
Motivation

1. We lost a lot of money.
2. We did not detect it in time.
3. We could have detected it in time and provided a fix.

Motivation

We need to make sure that the distributions of
input features are (always) the same
as in training.

MONITORING

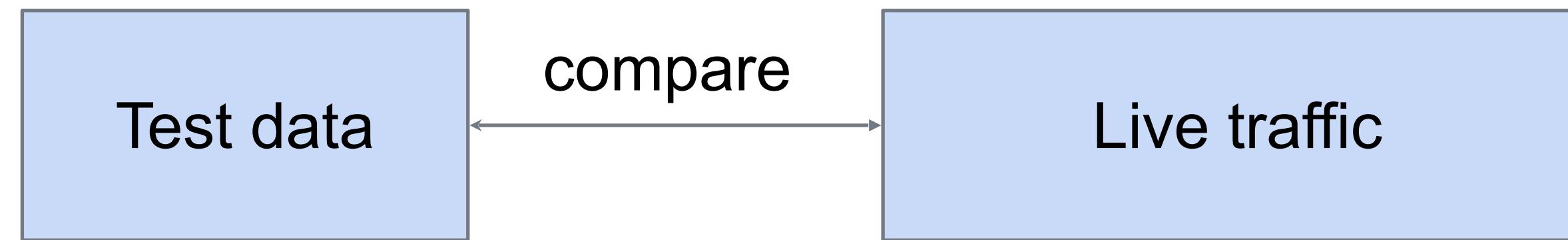


Make sure everything that runs live
is working as expected
→ Online world

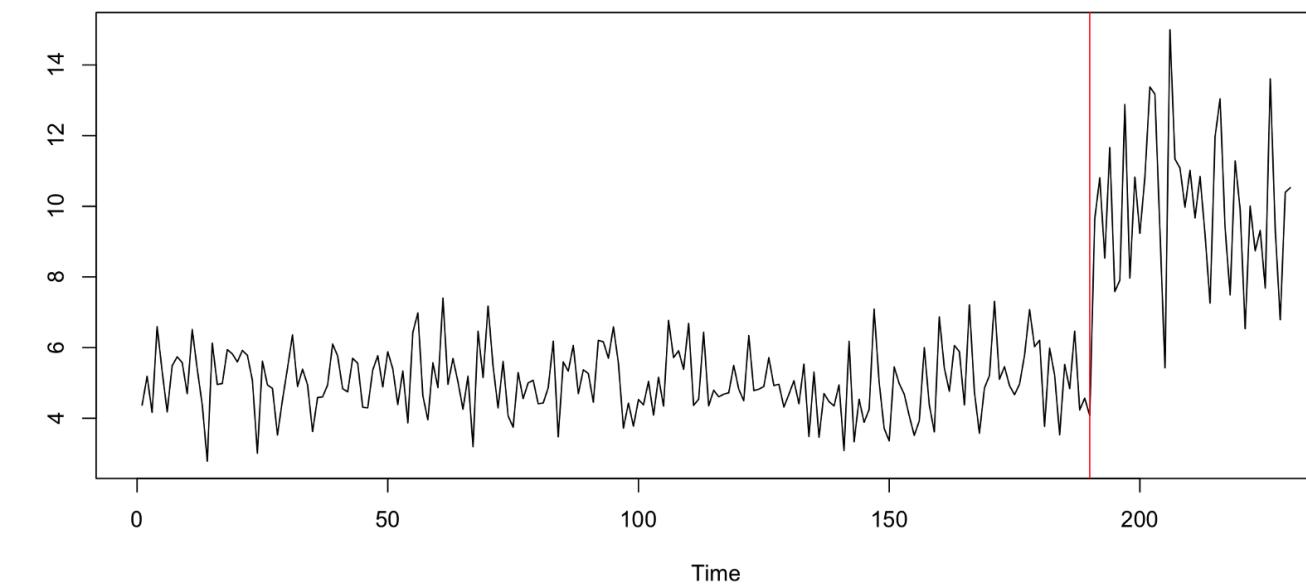
QUALITY MONITORING

There are two stages for model monitoring:

1. Check how models behaves once it is live.



2. Check if model is working continuously in live.



Monitor model with change point detection

QUALITY MONITORING

1. Check if model is working **before** it goes live
(typically done in a of “shadow mode”)

Compare live data distribution to test data:

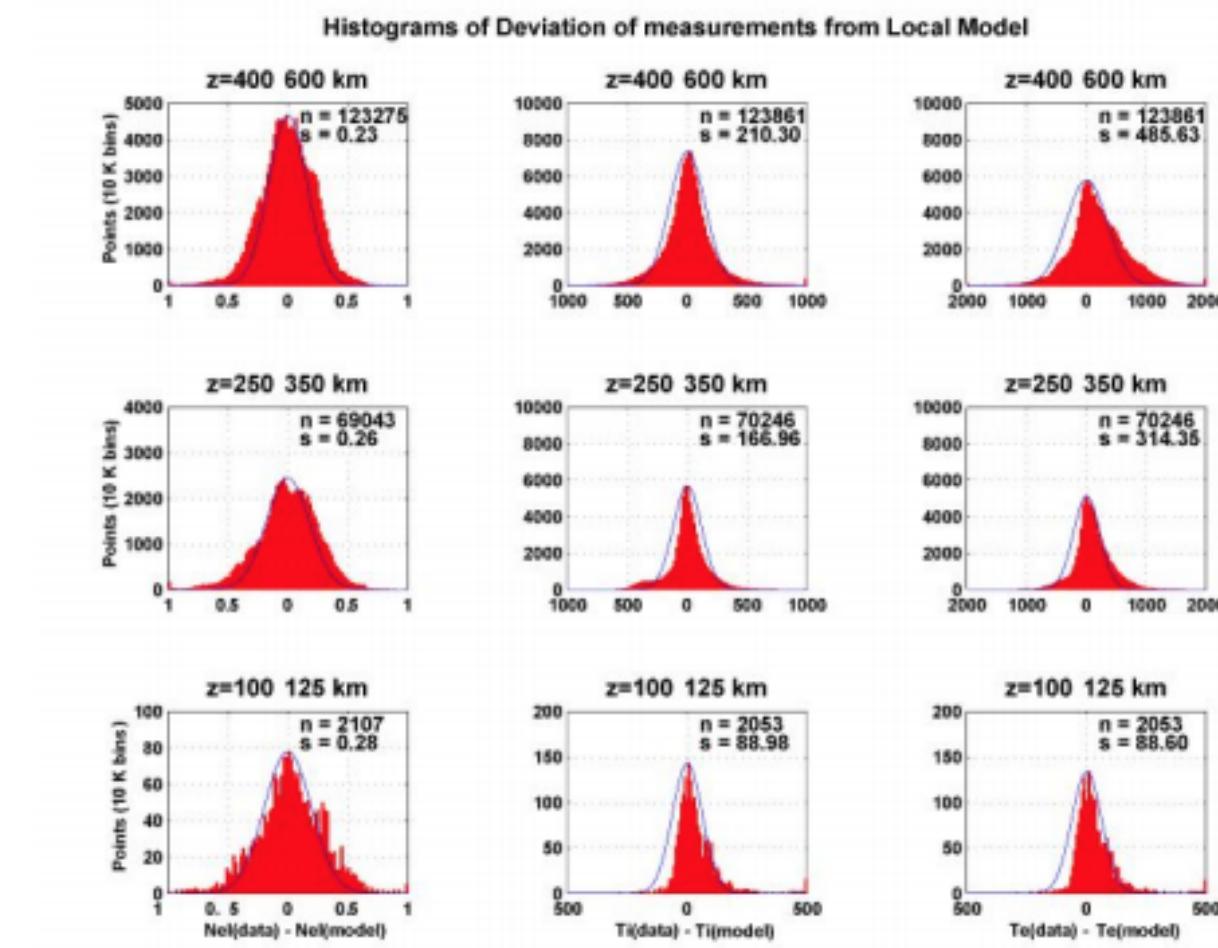
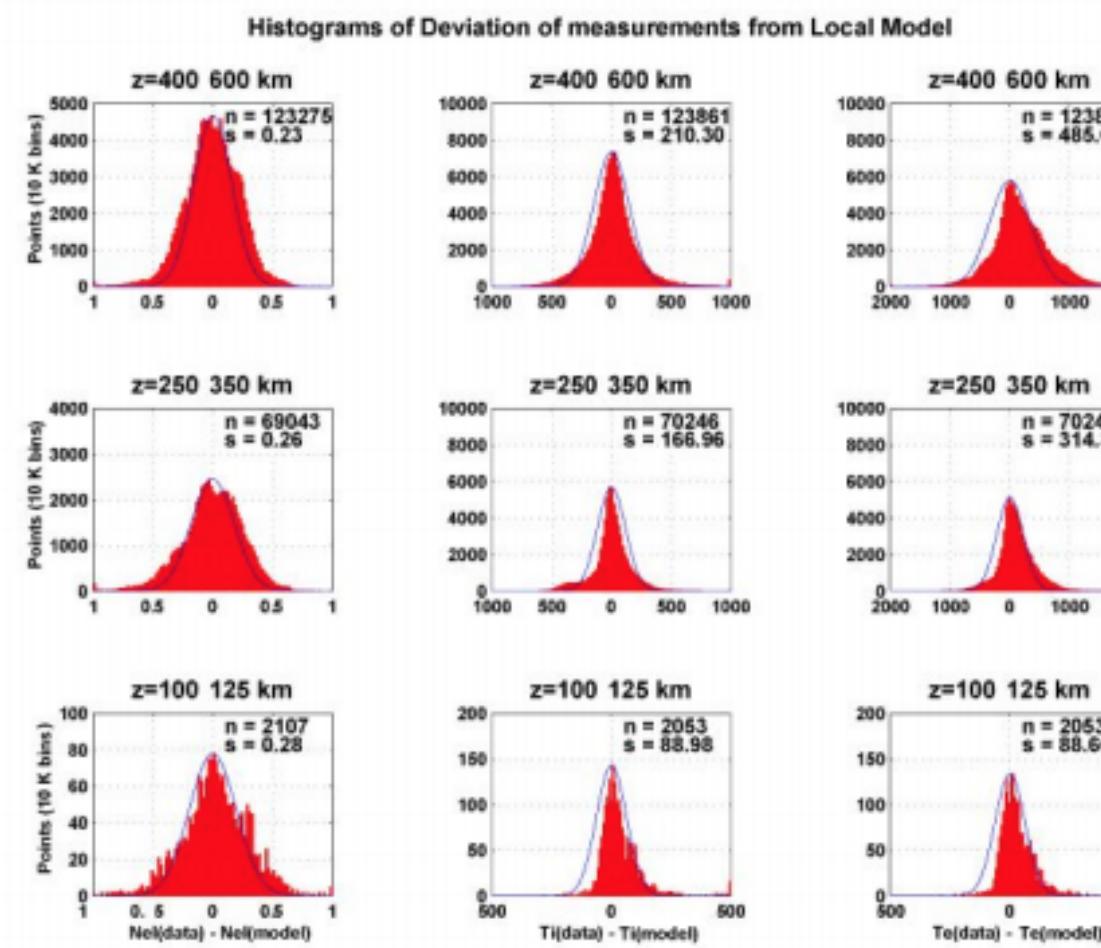
- Failing features
- Feature distribution
- Probability distribution

When everything looks good, put model live.

QUALITY MONITORING

Compare feature distributions and output probability:

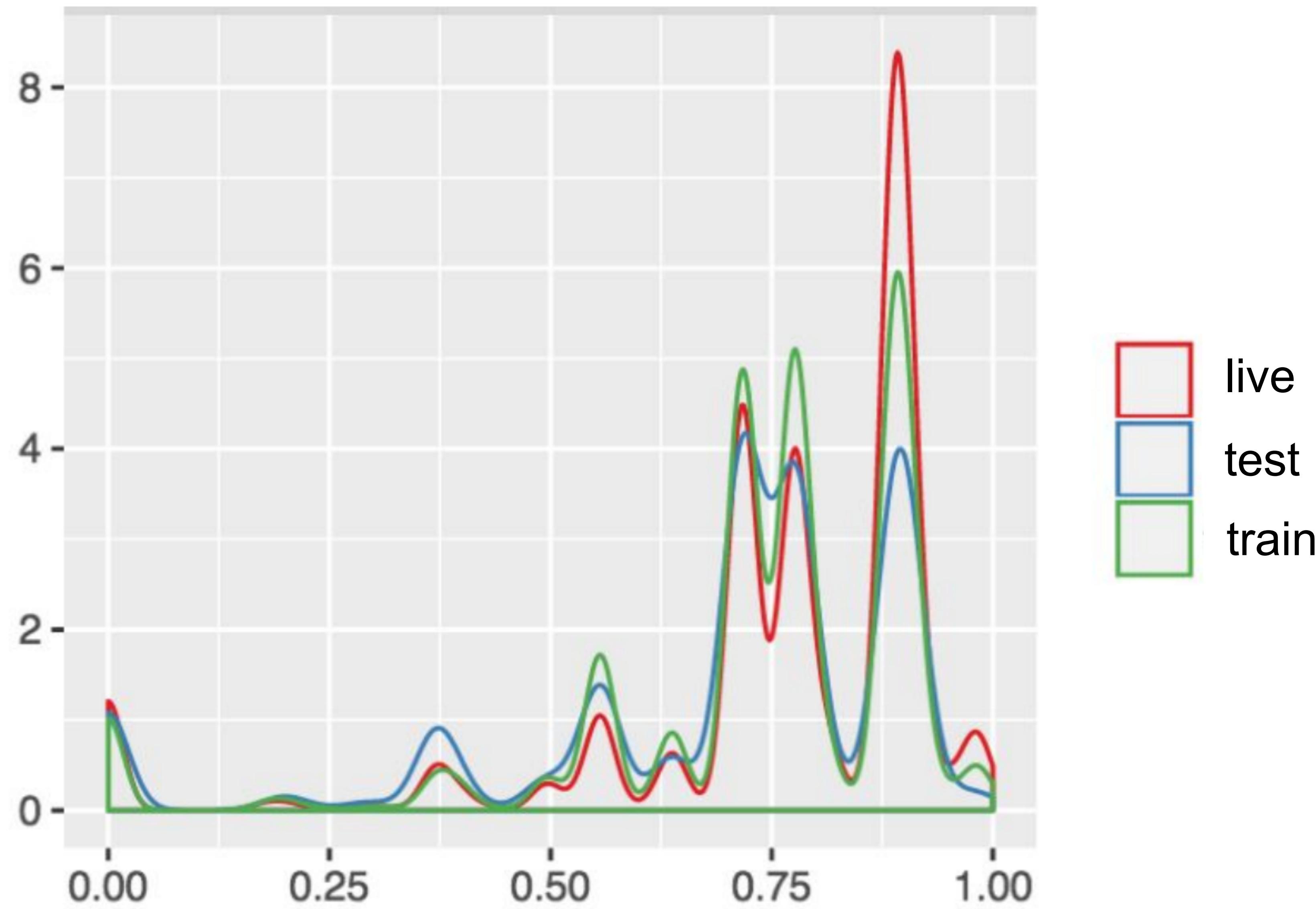
Feature distribution on test data



Quality Monitor

Feature distribution on live data

QUALITY MONITORING



Distribution Distance

How do we measure the distance between two distributions?

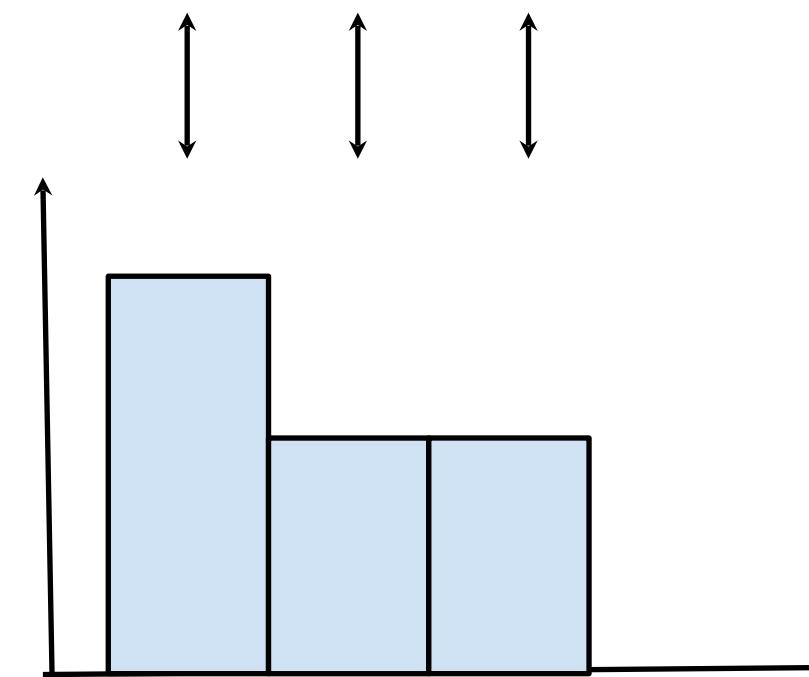
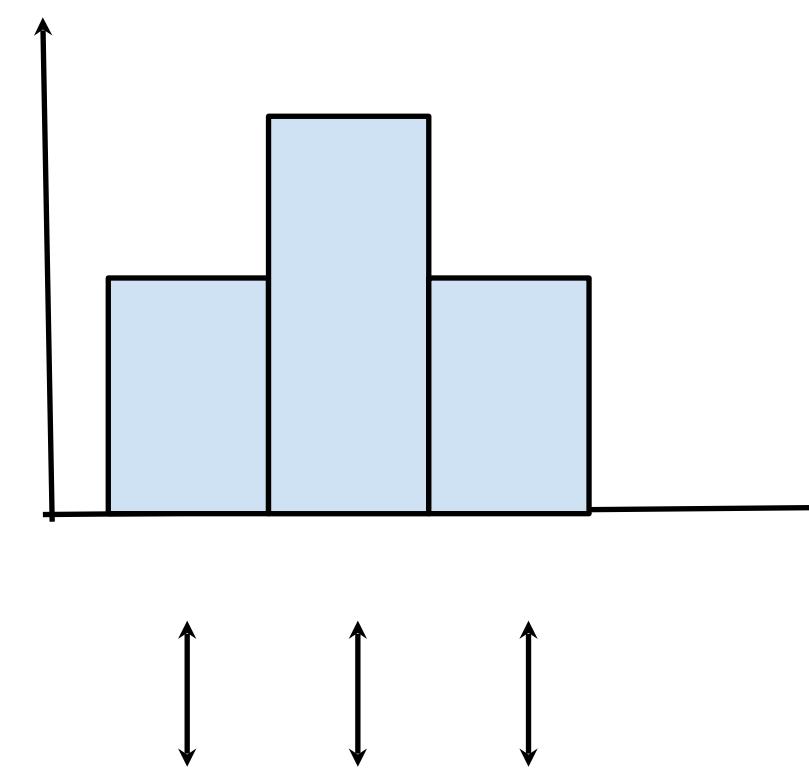
Answer: There are statistical test for that!

1. Kolmogorow-Smirnow-Test
2. Kullback–Leibler divergence
3. Wasserstein distance

Distribution Distance

Or: use a simple distance measure!

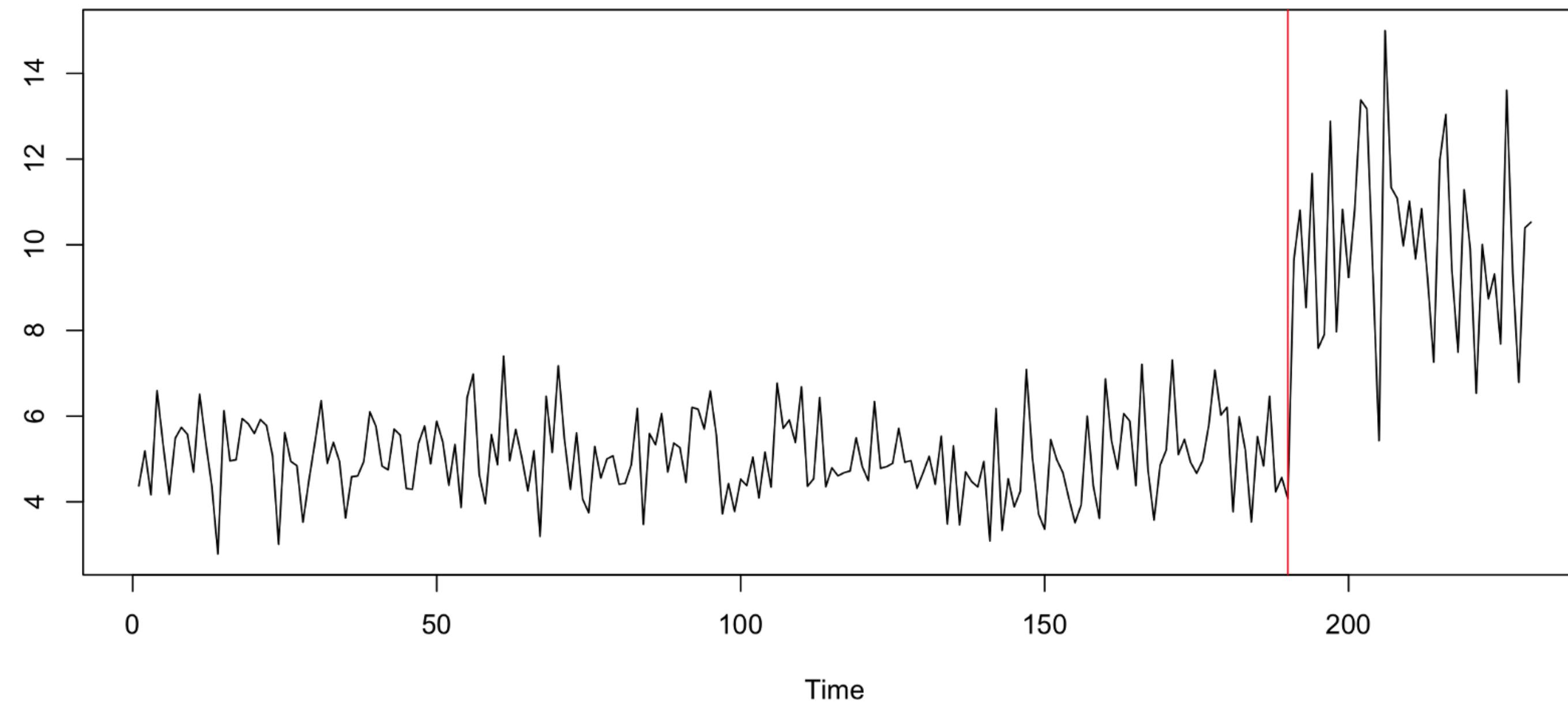
- Build a normalized histogram (divide each bucket by number of values)
- Count differences for each bucket



QUALITY MONITORING

2. Check if model is working continuously in live.
→ Change point detection

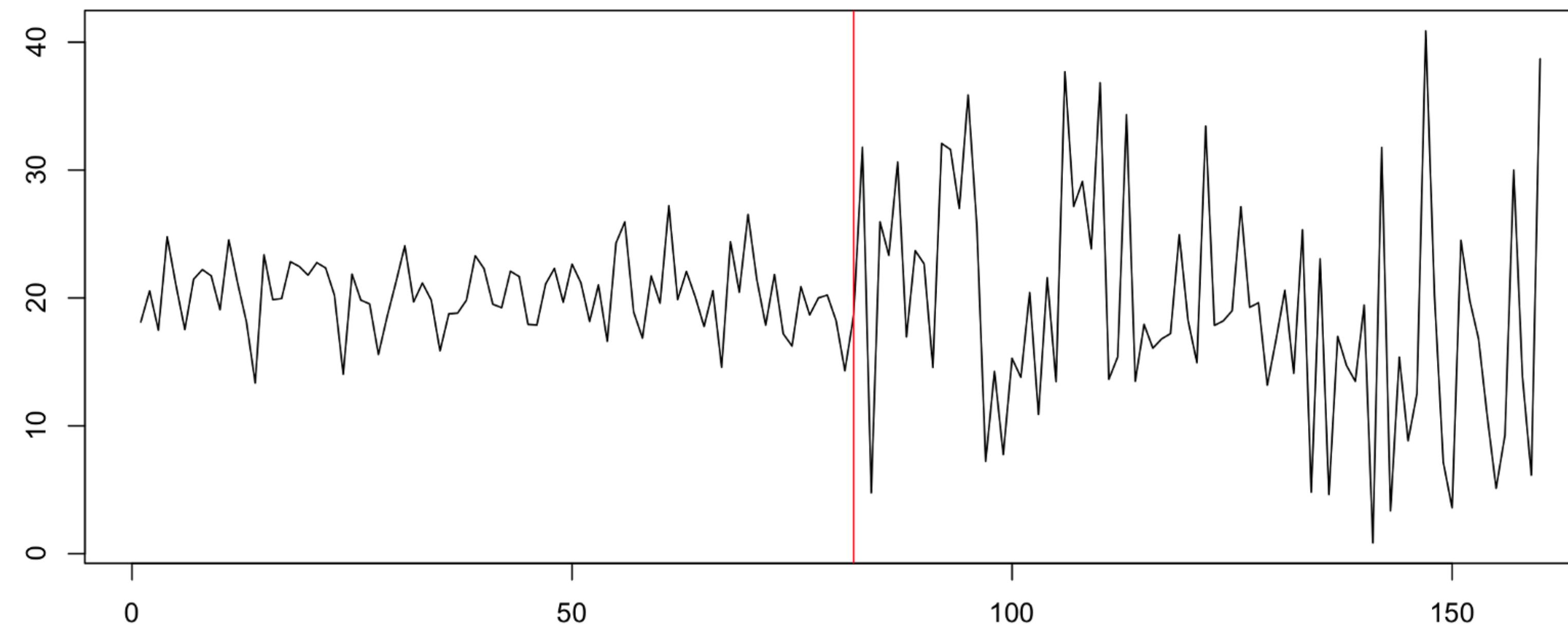
Idea: Check if a time series changes its behaviour at some point



Change in **mean** of time series

QUALITY MONITORING

Idea: Check if a time series changes its behaviour at some point



Change in **variance** of time series

QUALITY MONITORING

For every feature, we want to detect if such a thing happened.

There is a lot of research about this topic going on:

Reeves, Jack, et al. “A review and comparison of changepoint detection techniques for climate data.” Journal of Applied Meteorology and Climatology 46.6 (2007): 900–915.

Real world impact

Real world impact

Once we deploy a model, we somehow influence the state of the world:

- Who can return for free? (Return prediction)
- Who will receive a loan? (Risk prediction)
- Who will see a given ad? (Ad prediction)
- ...

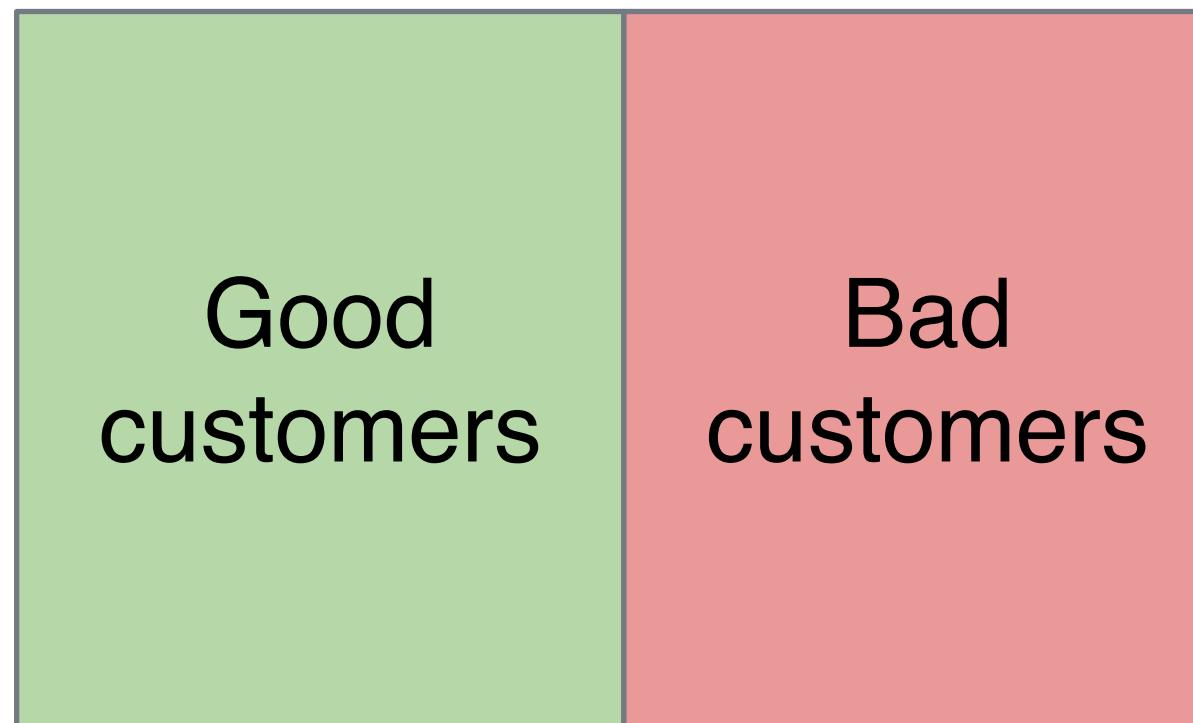
When this is the case, we face two problems:

1. Future training data will be biased.
2. We can run into a dangerous feedback cycle.

Real world impact

Problem 1: Bias of future training data.

Consider loan prediction (which customer do we offer a loan?).

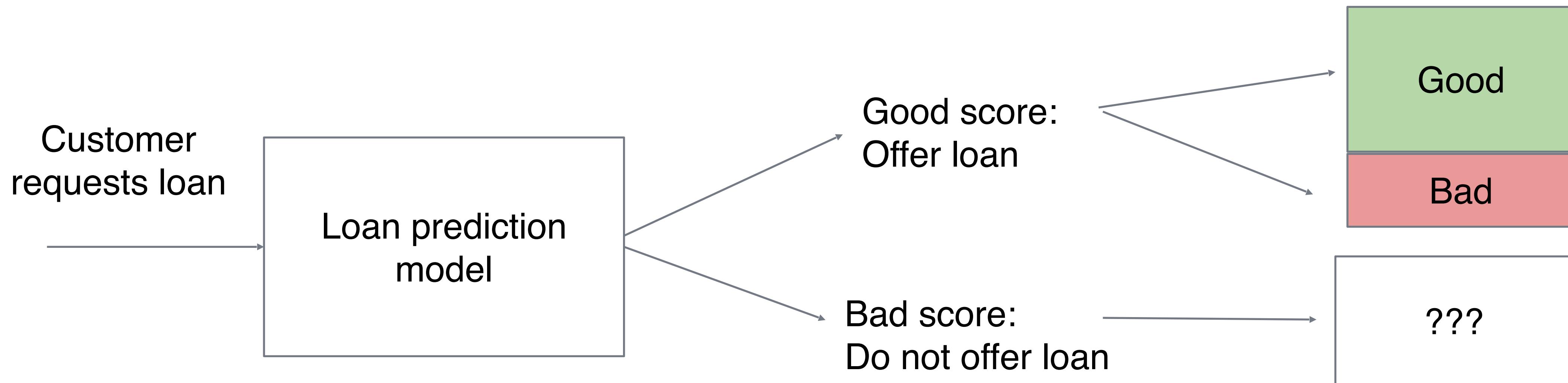


- We train a model which predicts for a given customer, that he/she pays back a loan.
- We have historic data that tells us, who did pay back in the past (good customer) and who did not (bad customers).

Real world impact

Problem 1: Bias of future training data.

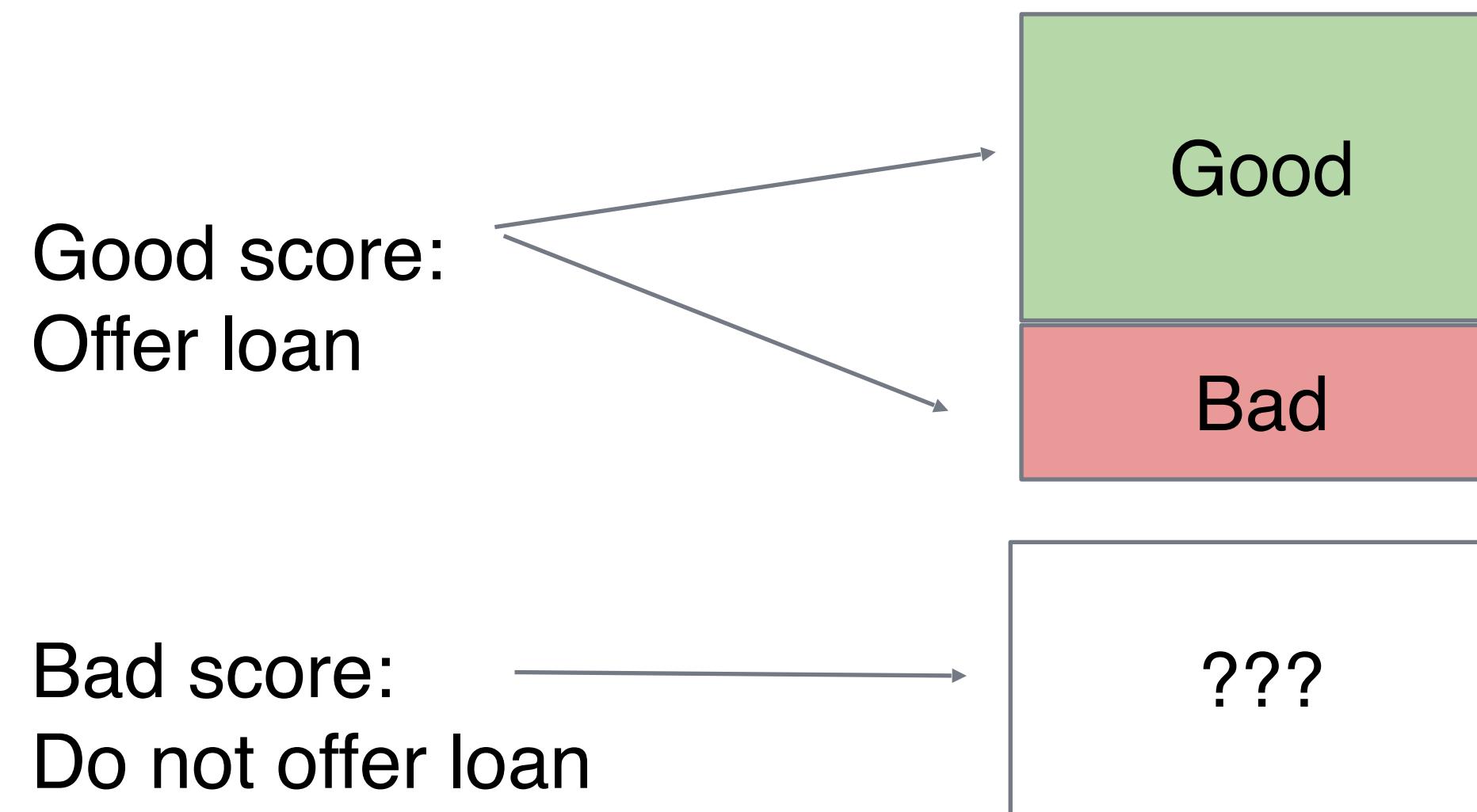
Consider loan prediction (which customer do we offer a loan?).



Real world impact

Problem 1: Bias of future training data.

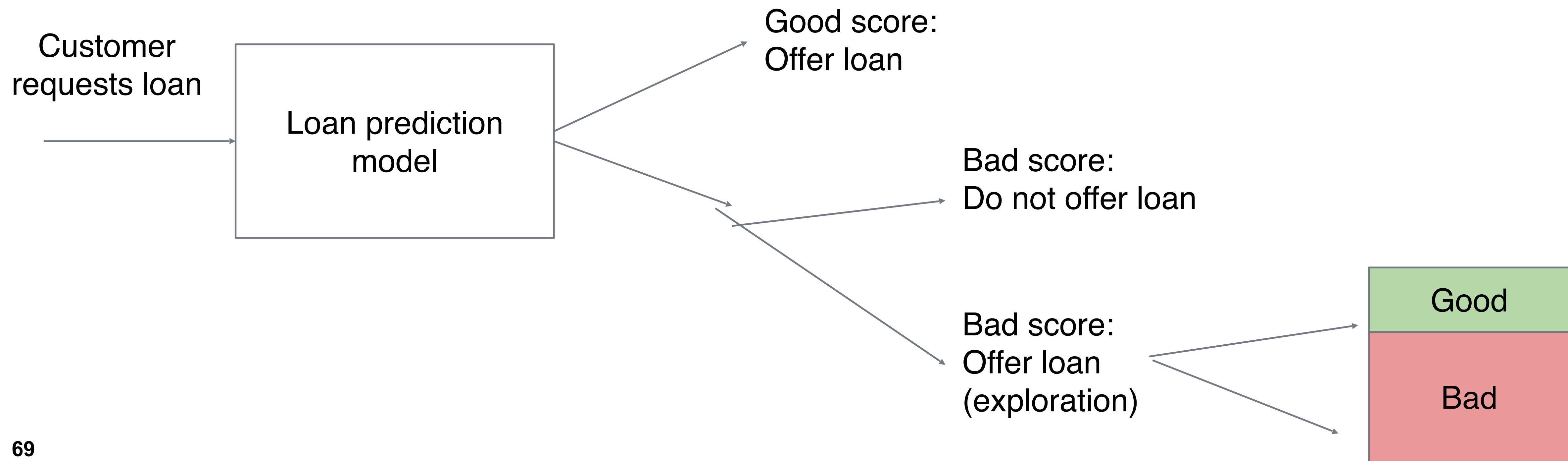
Consider loan prediction (which customer do we offer a loan?).



- We can observe the outcome, this can go into our future training data!
- We cannot observe the outcome, this cannot go into our future training data!
- But if we do not include this, we cannot include the cases from above, because our training data will then be seriously biased.

Real world impact

The only solution to this dilemma is to introduce “exploration” in our decision system. This means that we are willing to accept some cases in which the model gives a bad score.



Real world impact

How much exploration needs to be done is difficult to answer:

- No exploration: No future training data, but we more immediate profit.
- More exploration: More training data, less immediate profit.

To find the best ratio for exploration is really hard to do analytically (if not impossible).

Many people just do: “Let’s do 5% exploration”.

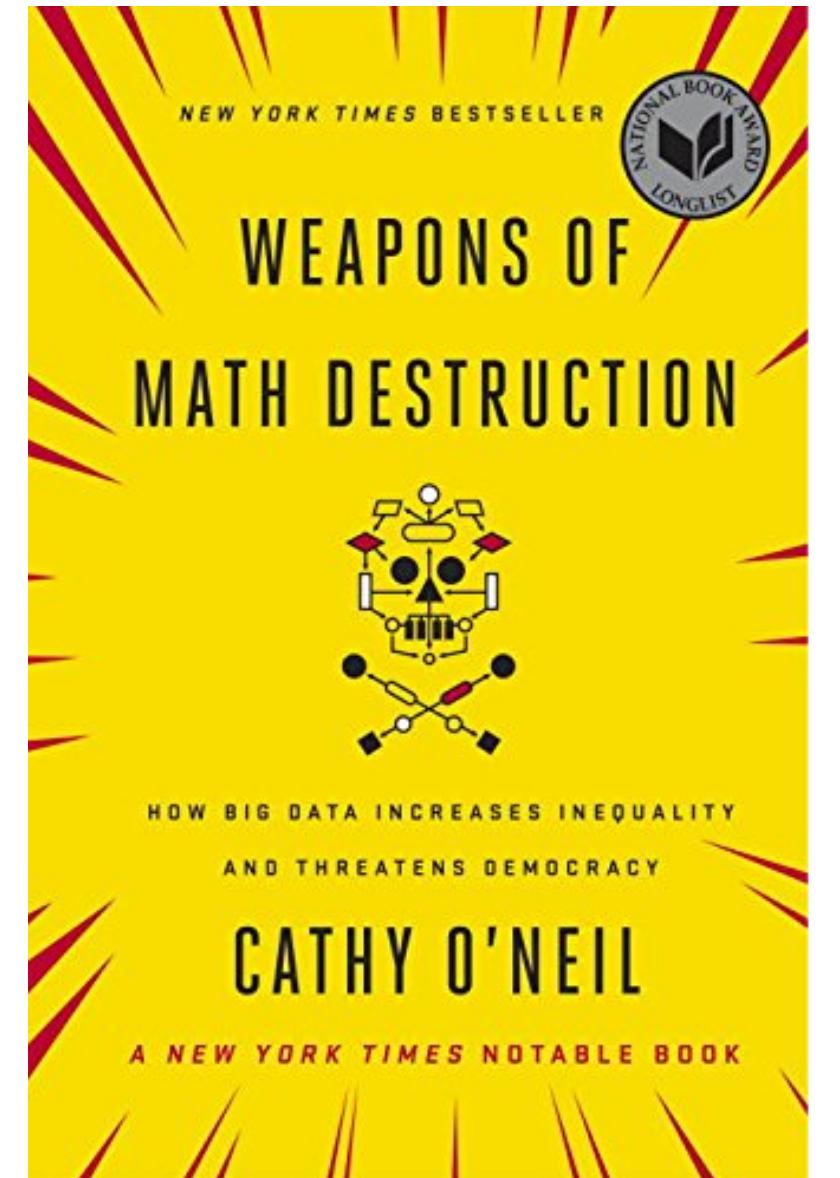
Real world impact

Problem 2: Feedback cycles

If we use the outcome of our models as labels, we run into a feedback cycle that empowers our models decision.

Example (from the book on the right):

- In the US, they use a survey to ask subjects questions like “was there crime in your neighborhood”. The outcome of this survey is used to feed a prediction model that predicts, if the person is a criminal. The output of the model is available to the jury.
- The survey does not include attributes like gender or race.



Real world impact

Problem 2: Feedback cycles

- People from poor neighborhoods will get a higher “crime-score” (because the model uses it as a feature).
- People from poor neighborhoods are more likely to be put into jail (because the jury reads the model output).
- When the model is re-trained, it will put even more weight on the feature “poor neighborhood”, because it saw more evidence, because more people are in jail.
- The problem is that the model uses its own outcome as training data.

Real world impact

Problem 2: Feedback cycles

- You can basically avoid this, by not using the output of your model as training data (as we have seen before), unless you do exploration.
- Problem solved? No!
- Sometimes it is not directly visible to you, how the outcome of your model influences the world (in this case through the jury).
- It could even be that people apply the output of your model without you even knowing (they read the score in some database) and then they probably do that without exploration!
- Be aware of this, be careful, know the consumer of your model.

Real world impact

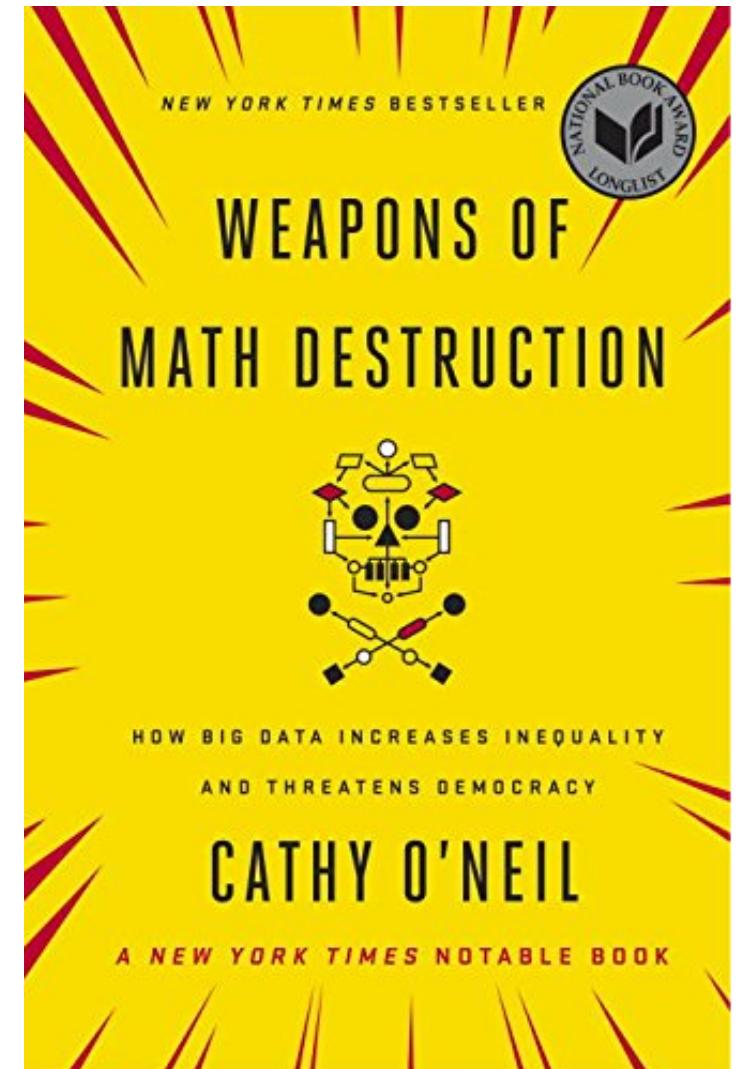
O’Neil warns in her book about this kind of abusive models.

She defines bad models as “weapons of math destruction”, if they fulfill the following criterias:

1. The user is not aware that a model is used in the background.
2. The model works against the interest of the user.
3. The model is not public, i.e. “intellectual property”.
4. The model makes predictions on a massive scale, predicting on many users.

MI-models are often opaque, unquestioned, unaccountable

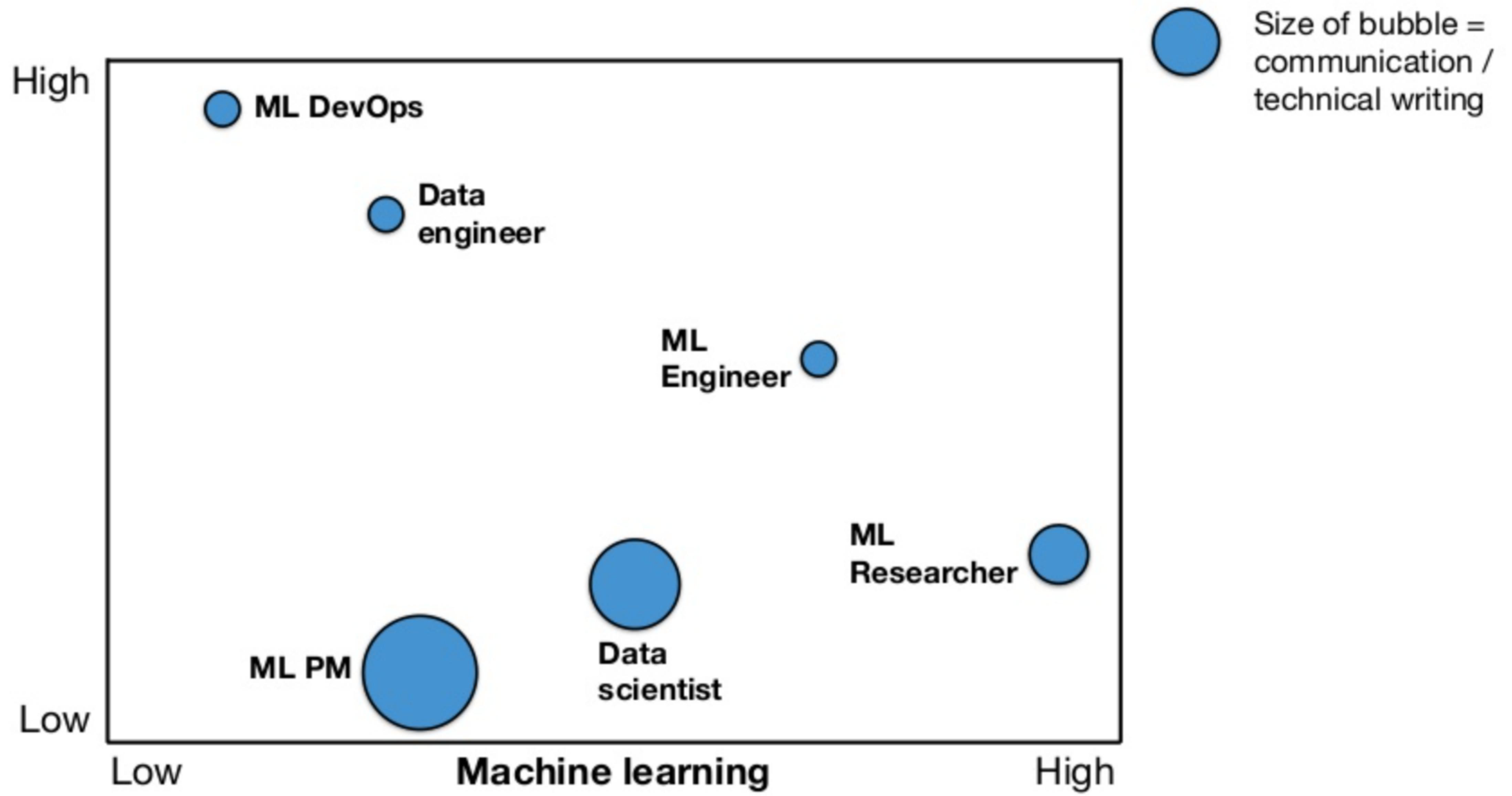
According to this definition most models nowadays are “bad” :(

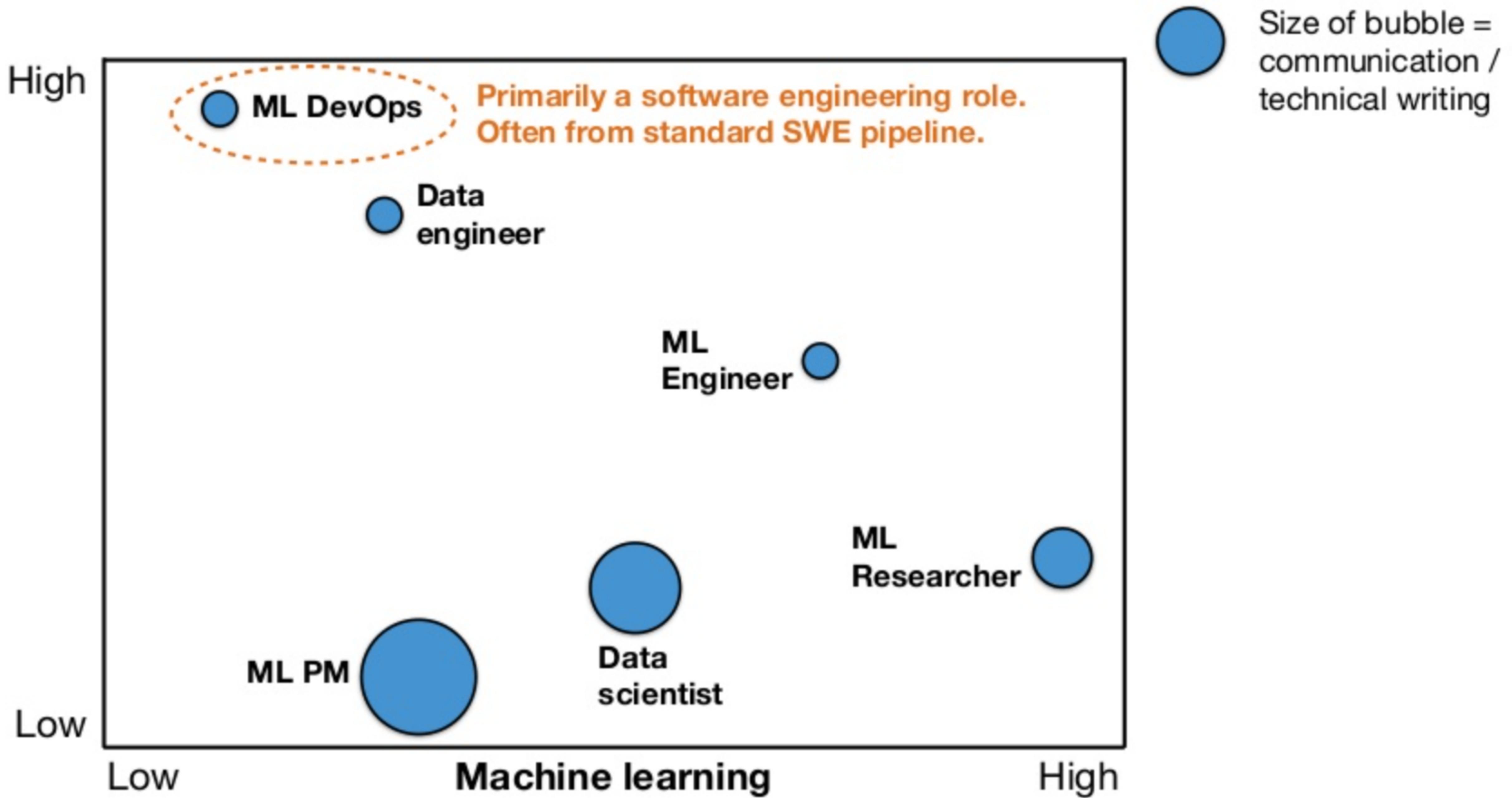


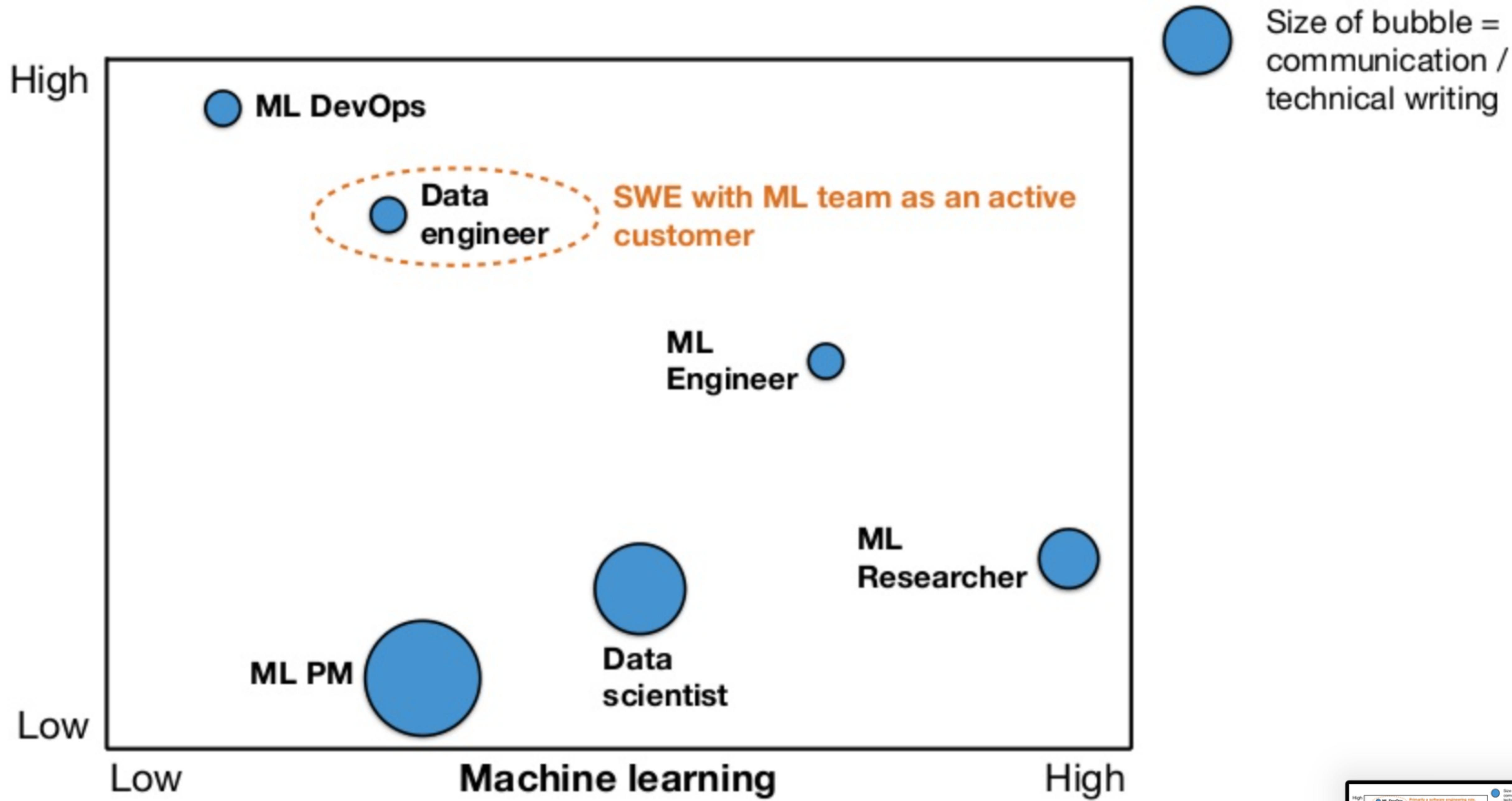
Data Science Organisation

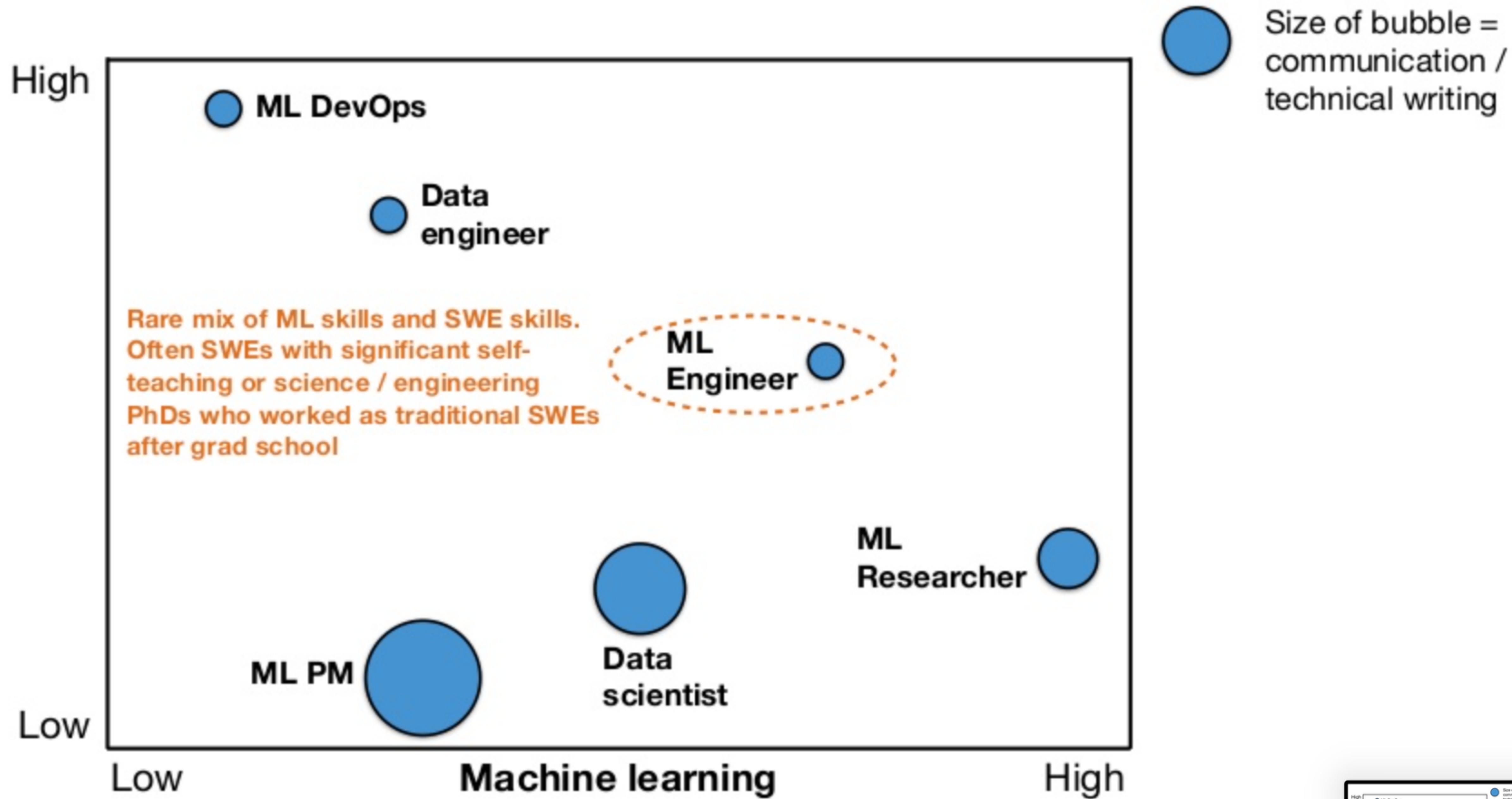
Disclaimer:

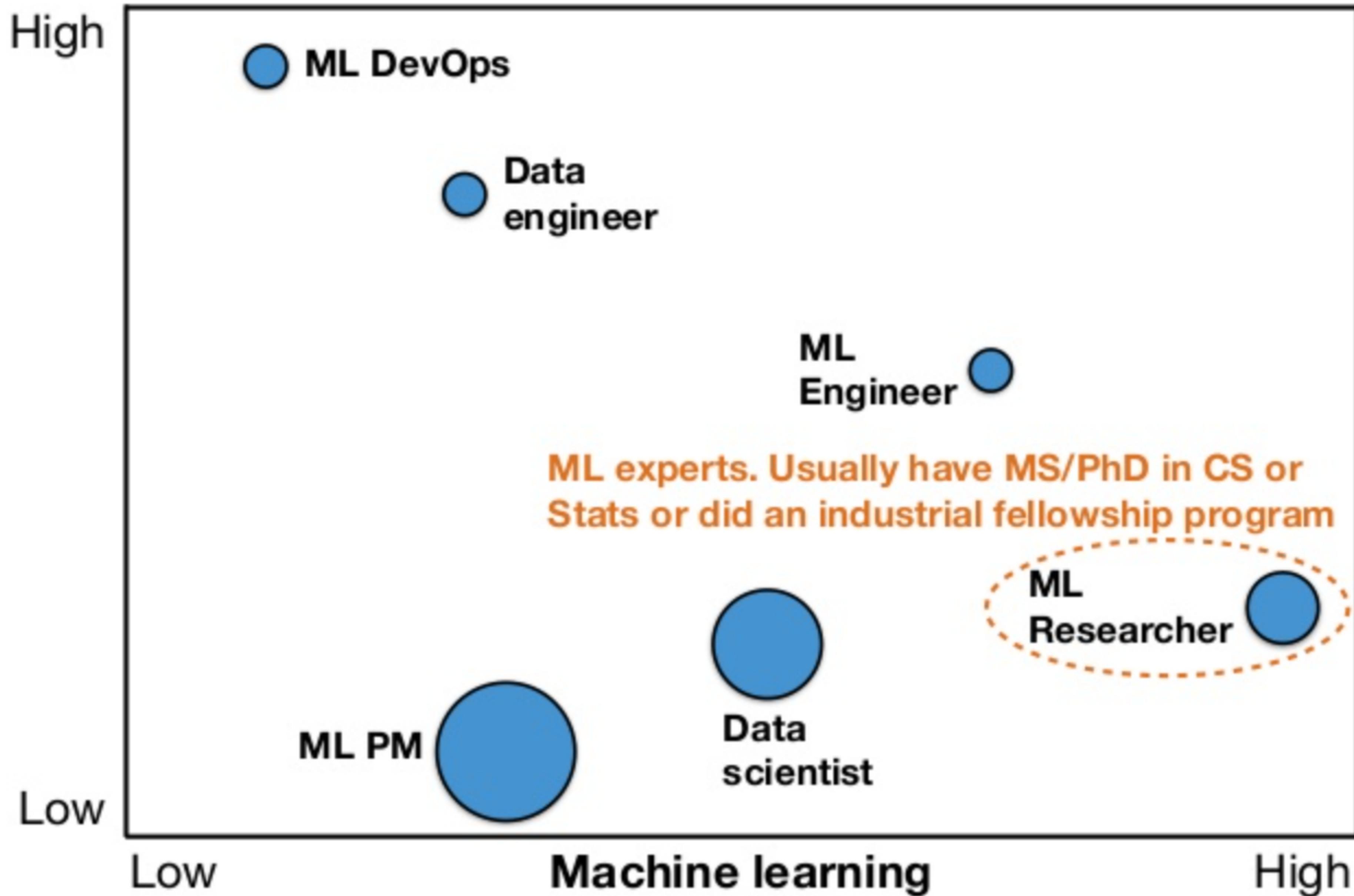
Unlike in more mature fields like Software Engineering (SE), there is no state-of-the-art DS organization form. Expect that people do not know how to handle your “ivory tower team” and try to apply the same methods as for SE. You need to educate them that DS is quite different.

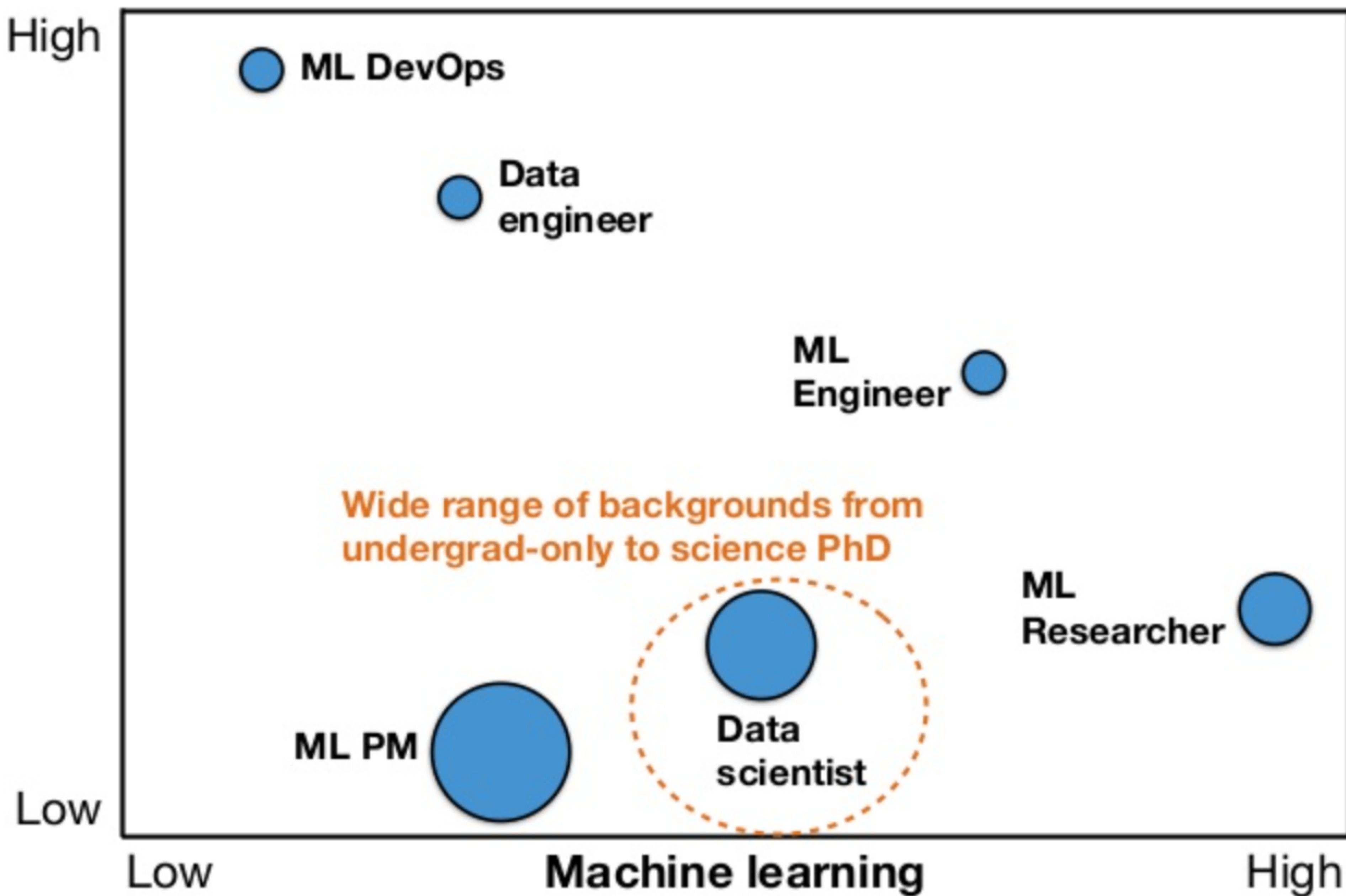


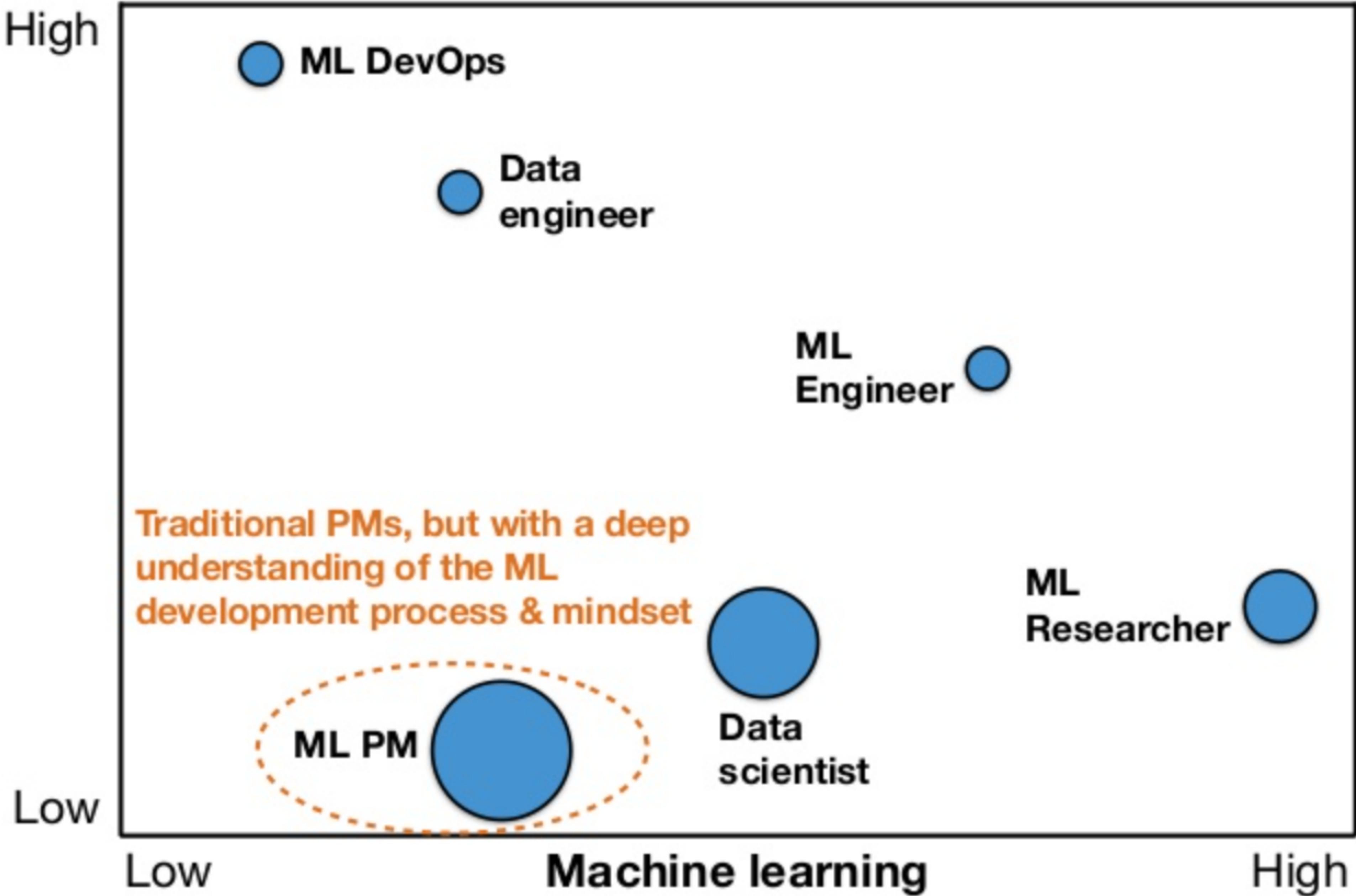






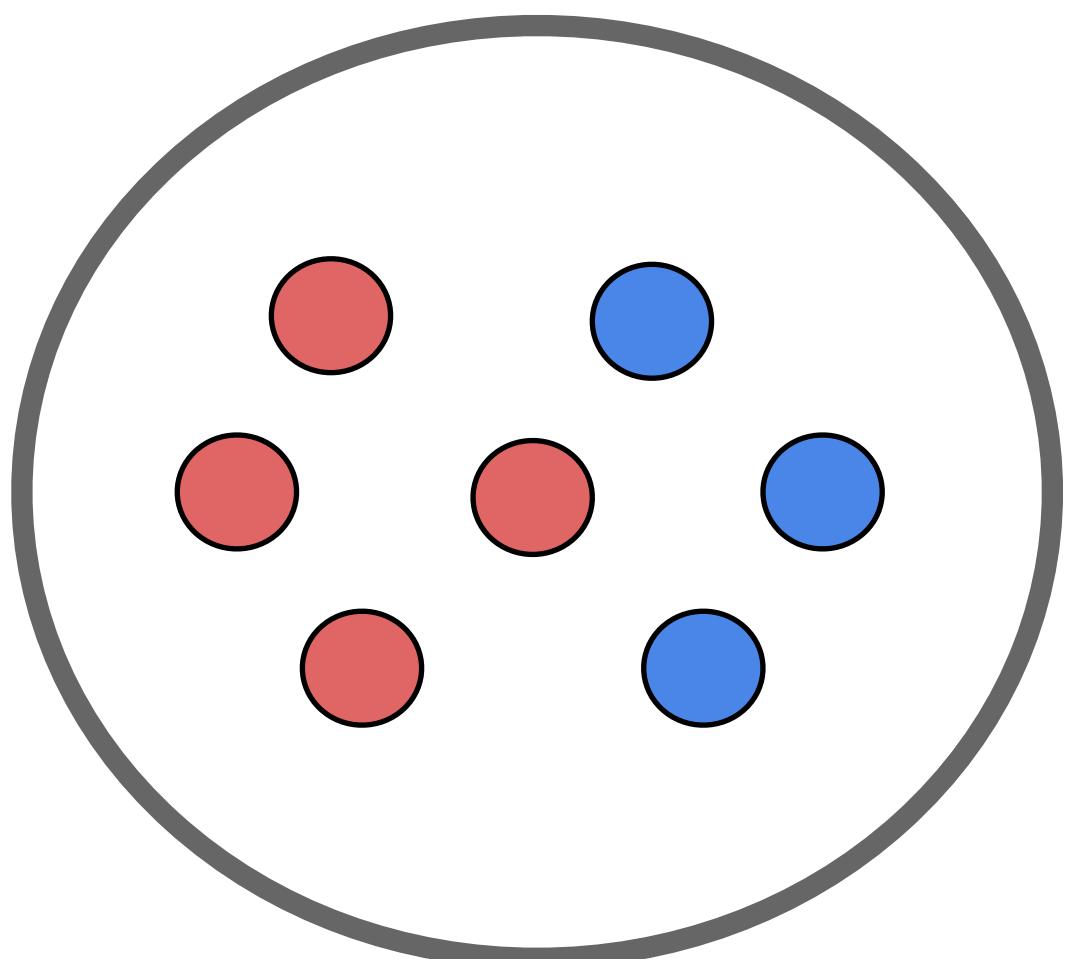




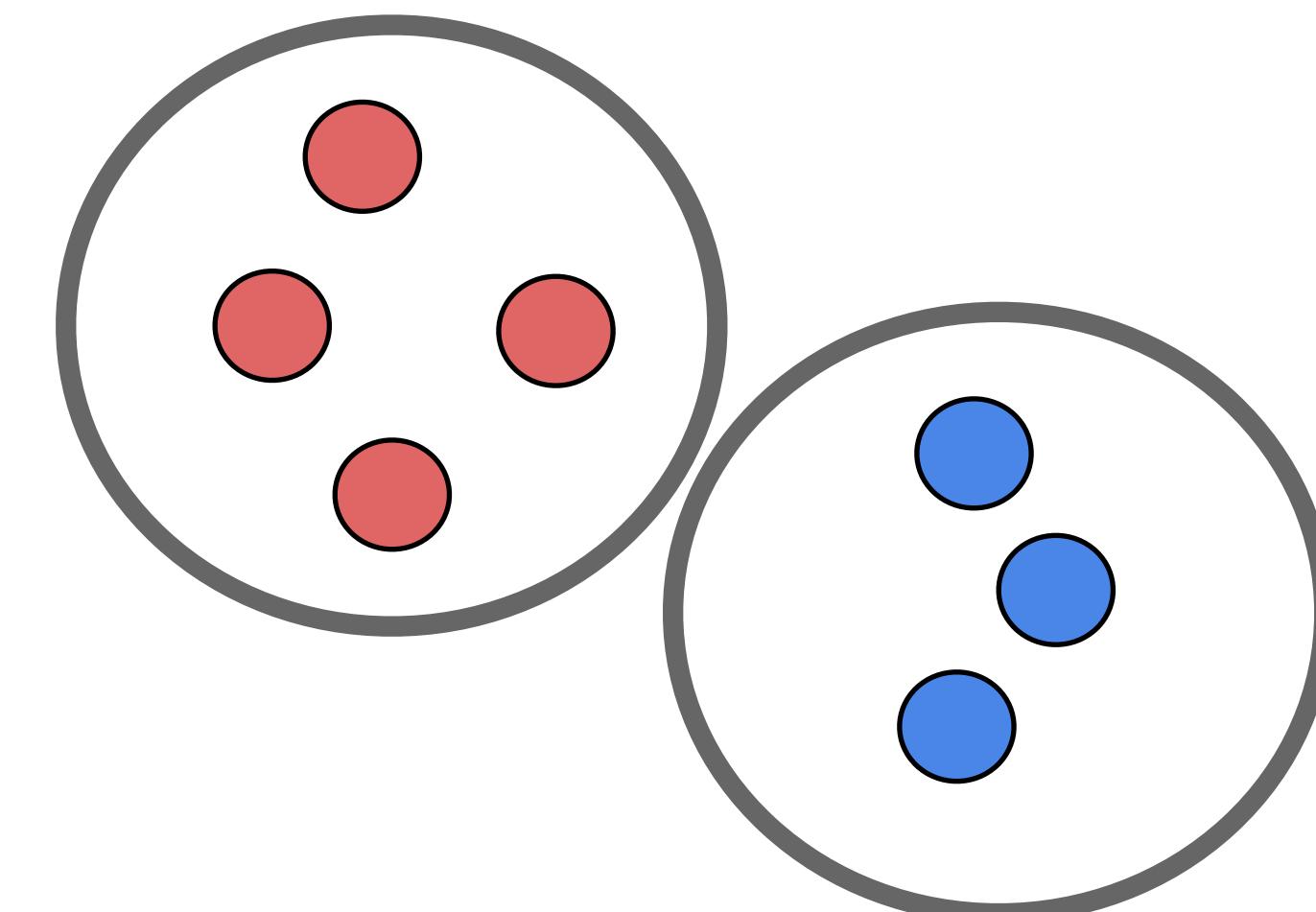


DS Team Structure

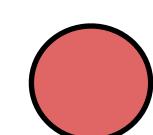
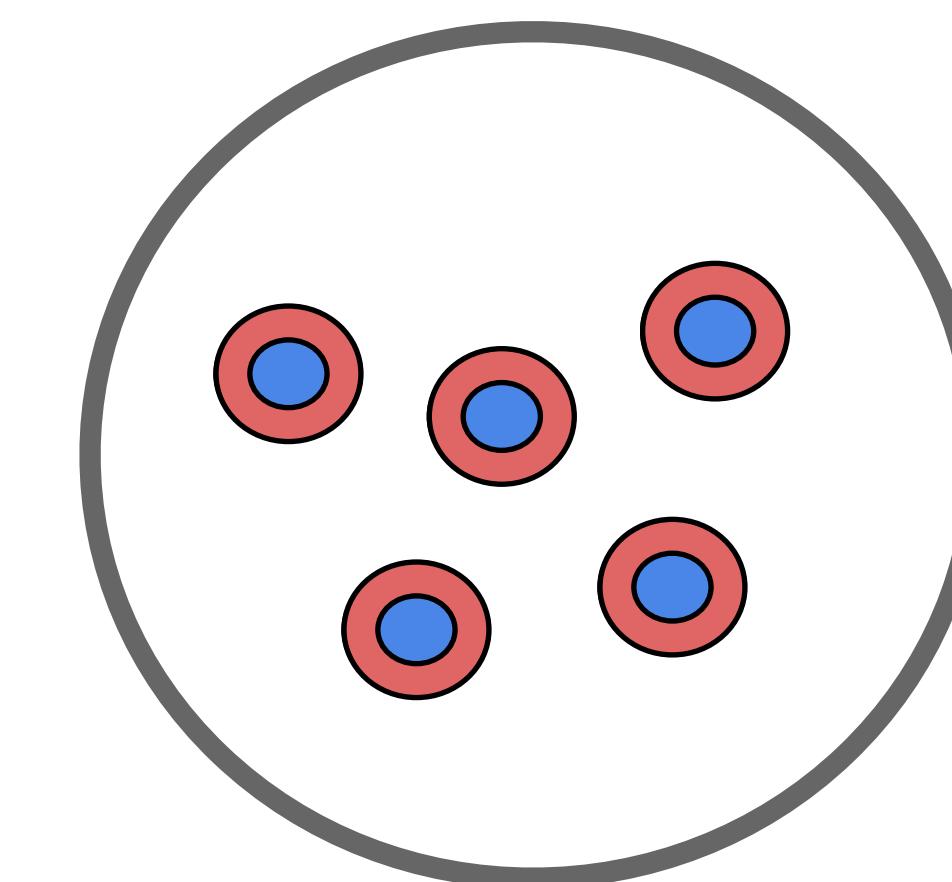
Mixed Team



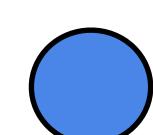
Separate Team



Hybrid Team

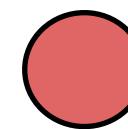
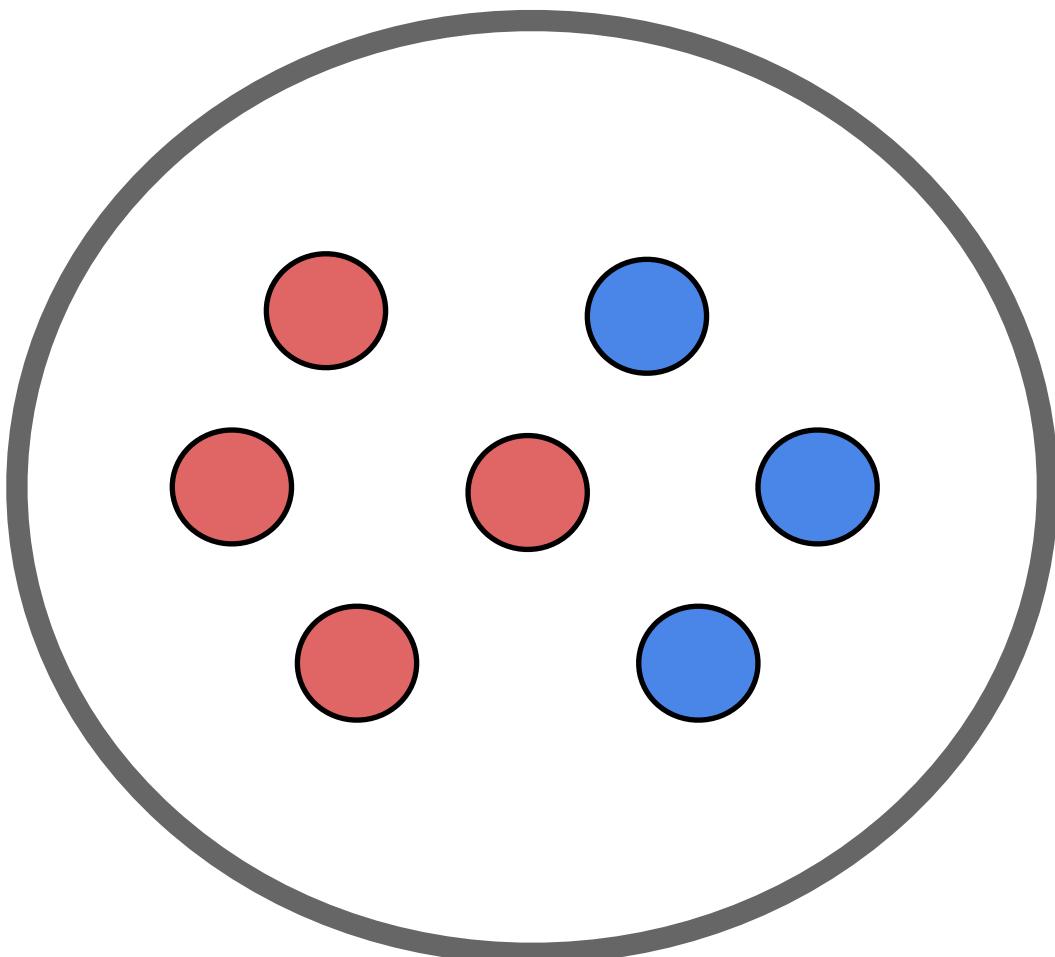


Data Scientist



Software Engineer

Mixed Team



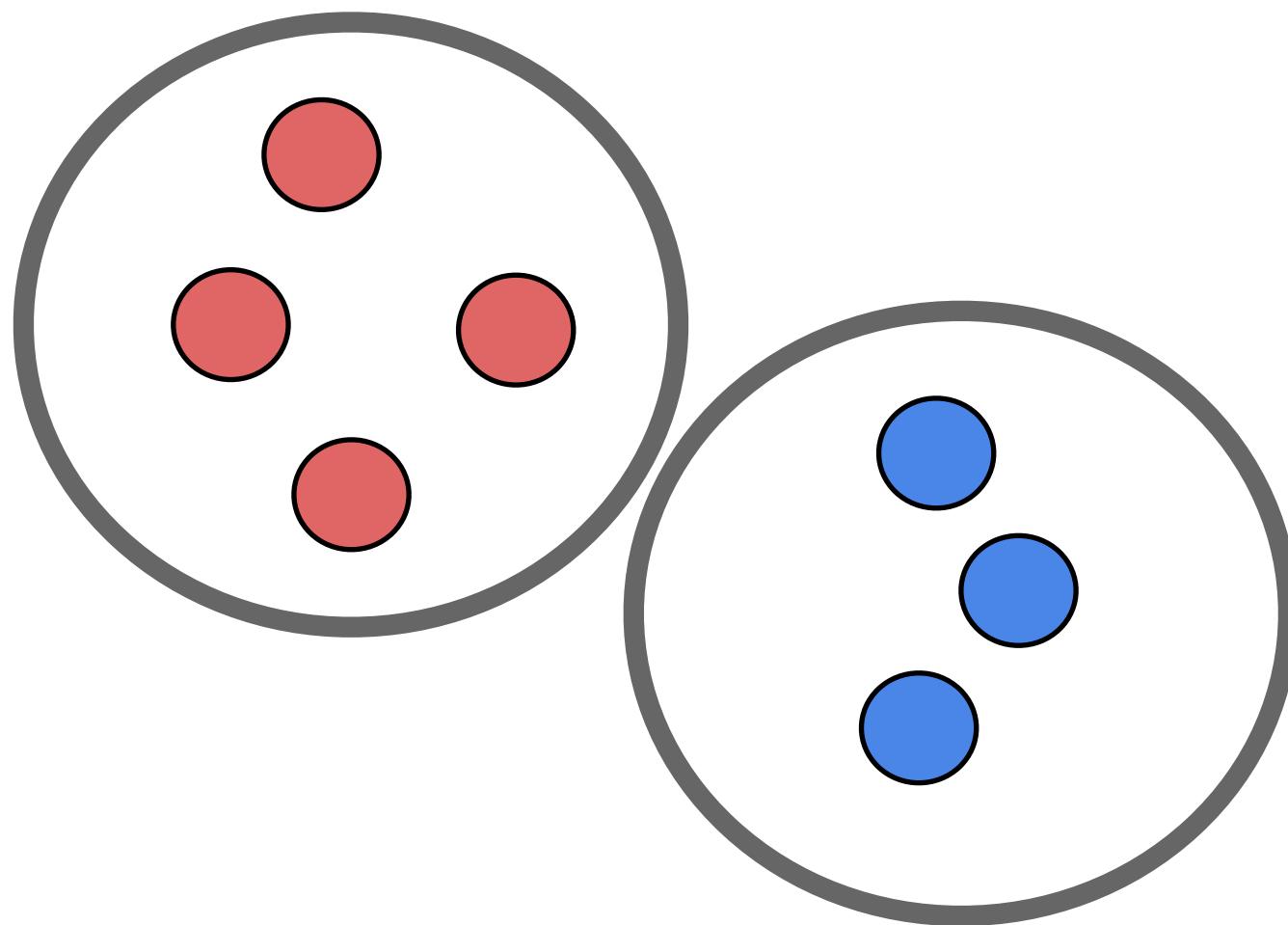
Data Scientist



Engineer

- All work together
- Engineers and DSs are equal
- Engineers:
 - Deployment
 - Automation
- Data Scientists:
 - Data Analysis
 - Experiments
- Both: Data Engineering

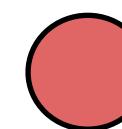
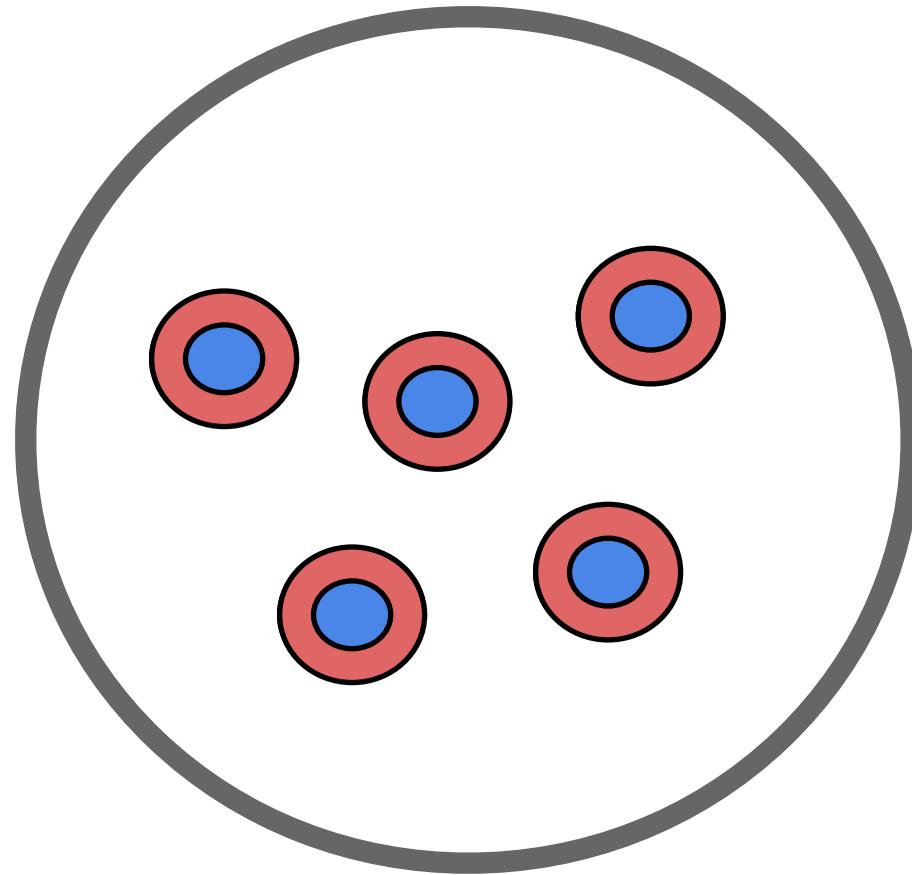
Separate Team



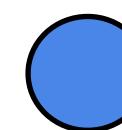
- “Throw over the fence” setup
- Working climate is often very difficult:
 - DSs see Engineers as code monkeys.
 - Engineers sees DSs as unproductive ivory tower people.
- If you work in such a setup: Make sure to talk and socialize with the engineers (show interest in their topic, go for lunch etc...)

- Data Scientist
- Engineer

Hybrid Team



Data Scientist

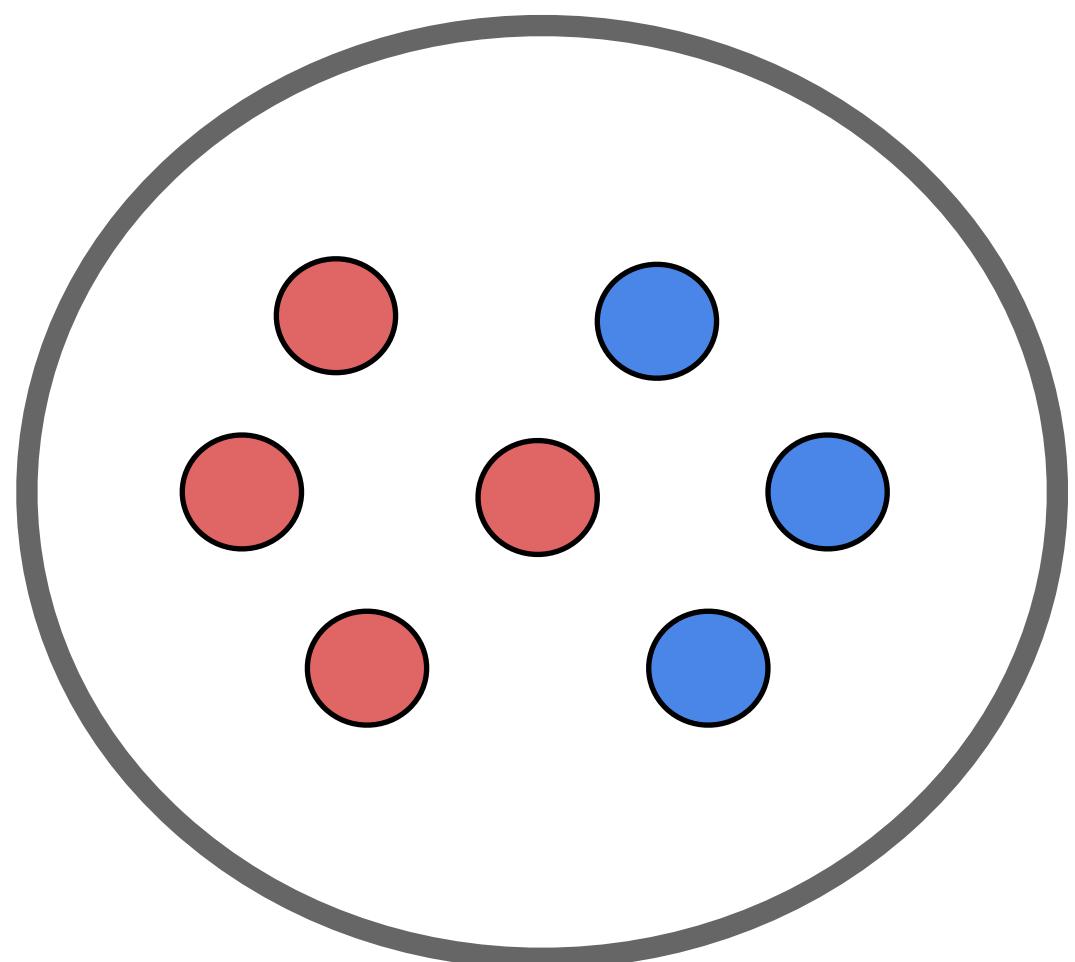


Engineer

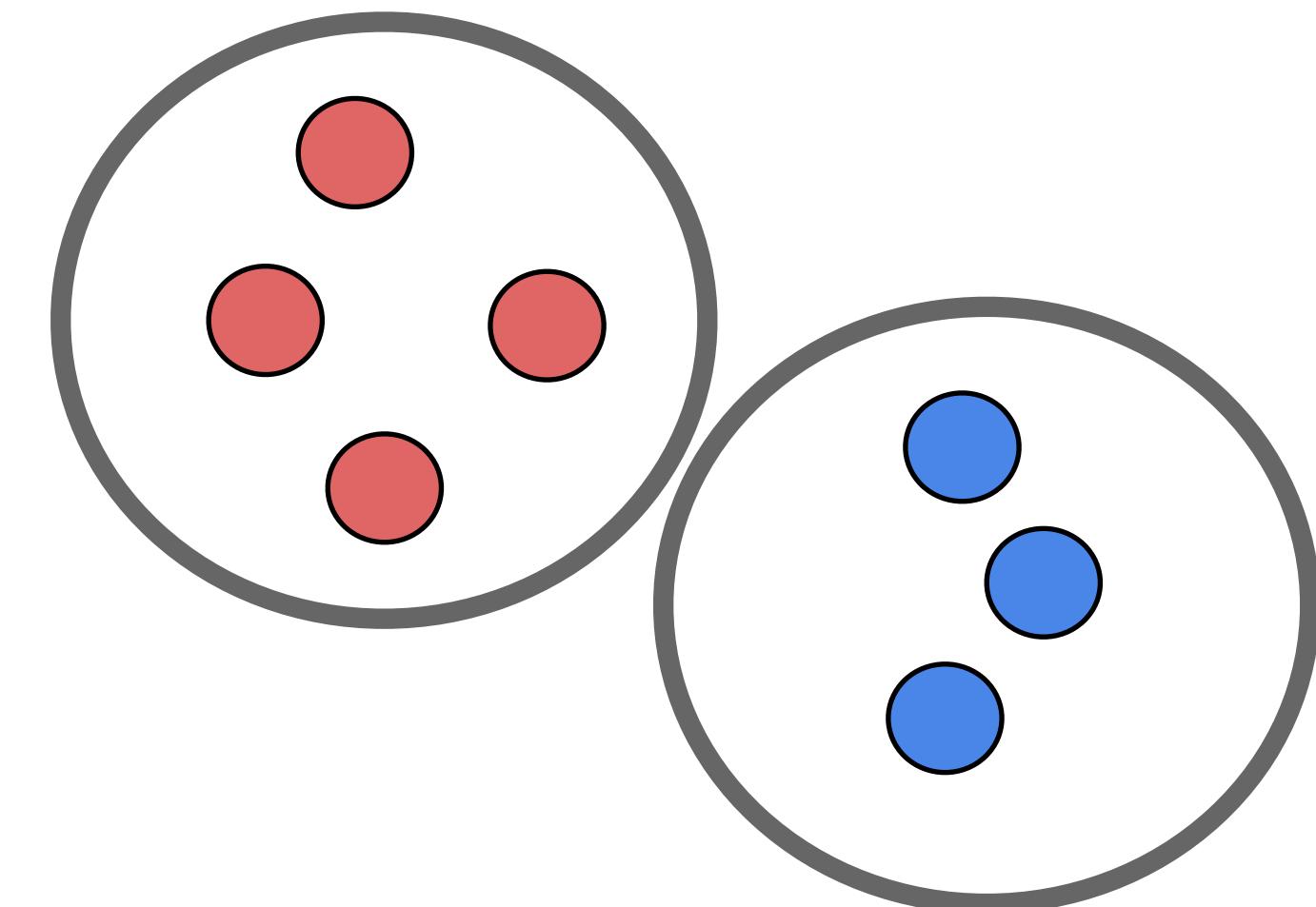
- People in this team do both: Engineering and DS.
- Often found in small start-ups.
- Gives you the chance to learn a lot about running a production system (a very valuable asset that will differentiate you from most other DSs).
- But: Make sure that you have still enough DS time as business grows.

DS Team Structure

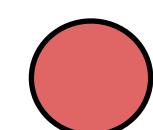
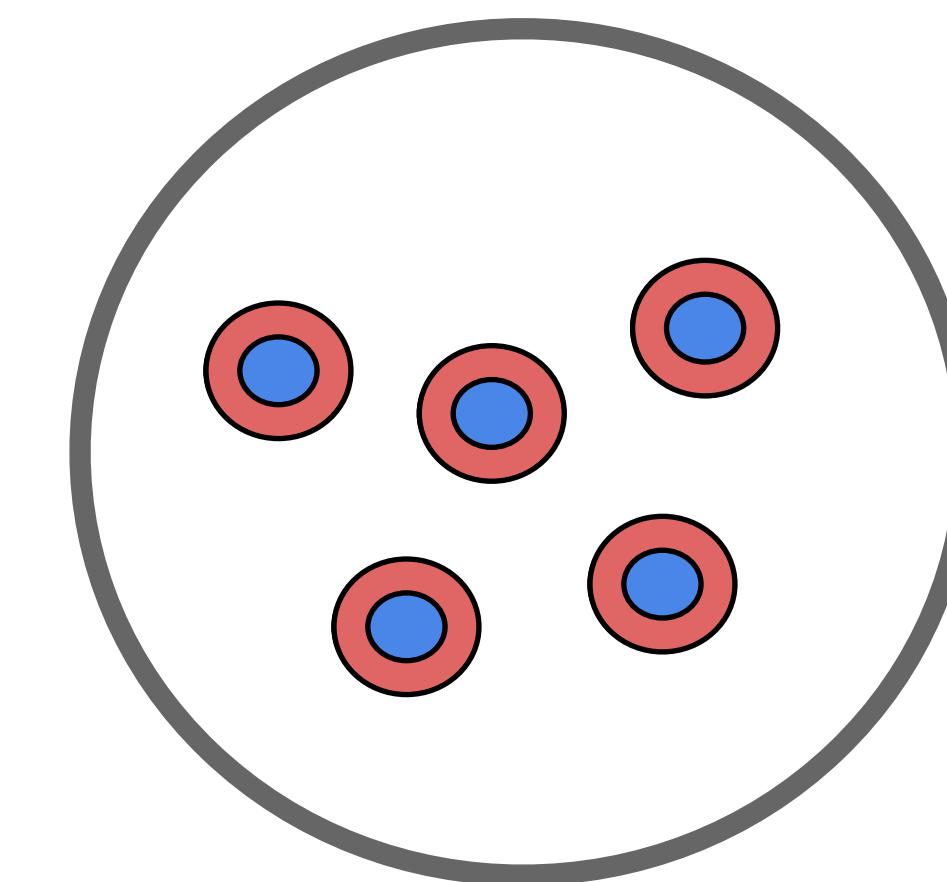
Mixed Team



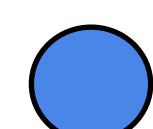
Separate Team



Hybrid Team

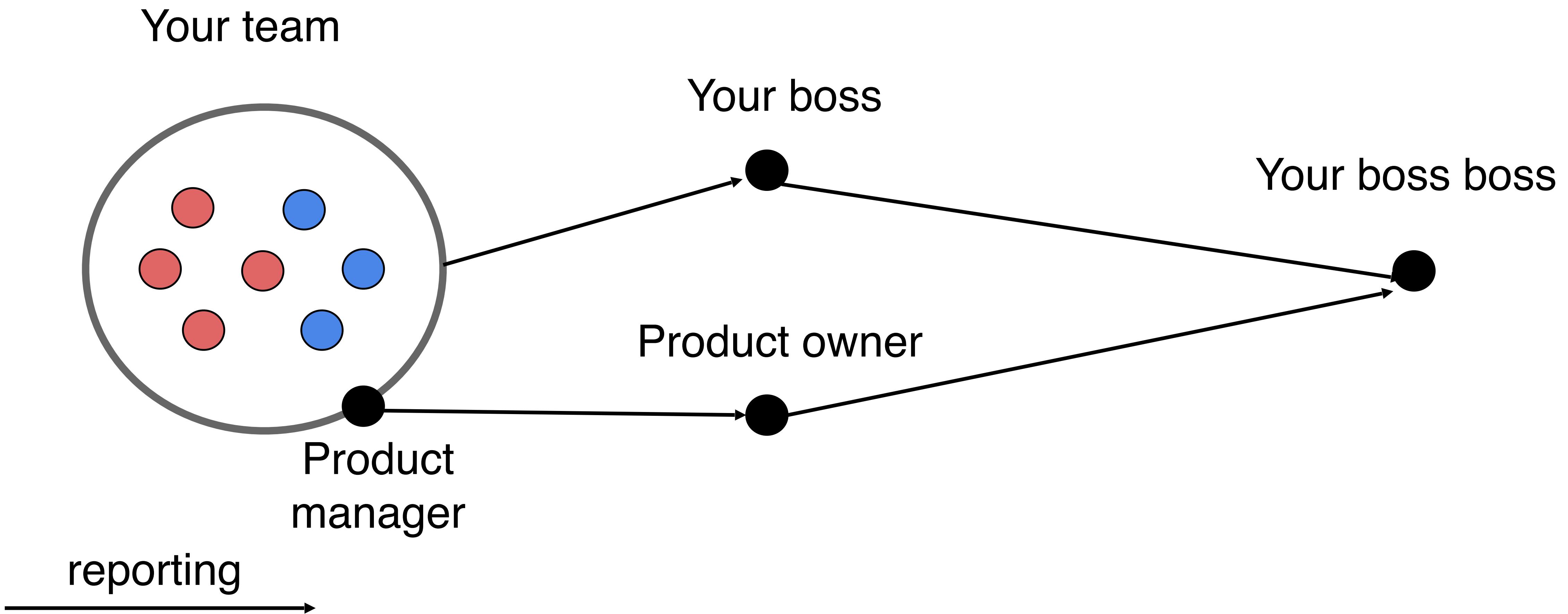


Data Scientist



Engineer

Your Stakeholders



Product Team

Product manager:

- Maybe within your team or not, maybe sitting with you or not.
- Breaks down requirements from product owner (tries to tell you what. you do, may creates tickets, may want to know what you do).

Product owner:

- Creates the product vision and communicates it.
- Manages external stakeholders for the product.

AI Product Management

How should PMs and AI teams work together? Here's one default split of responsibilities:

Product Manager (PM) responsibility

- Provide dev/test sets, ideally drawn from same distribution.
- Provide evaluation metric for learning algorithm (accuracy, F1, etc.)

AI Scientist/Engineer responsibility

- Acquire training data
- Develop system that does well according to the provided metric on the dev/test data.

This is a way for the PM to express what ML task they think will make the biggest difference to users.

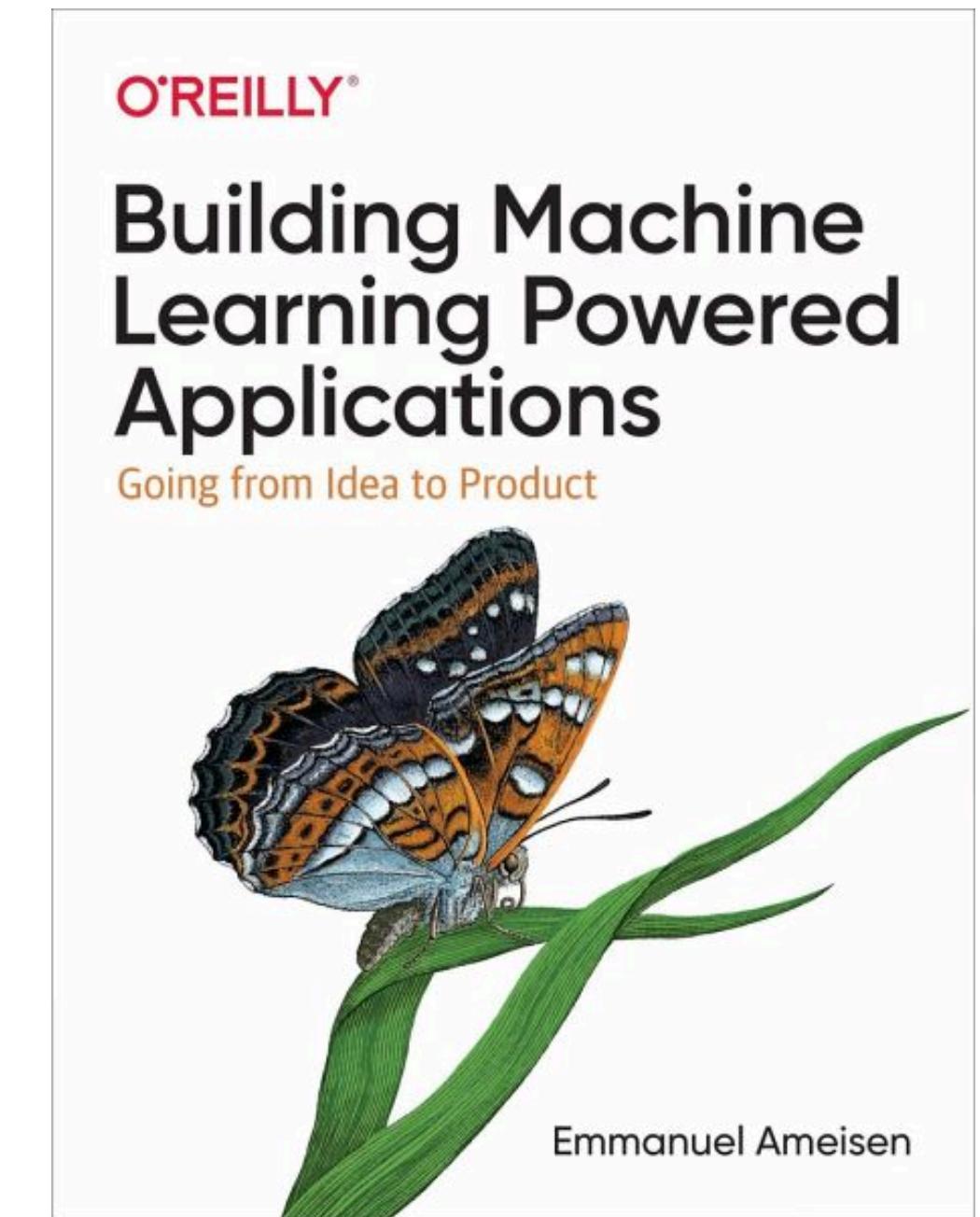
Your Team

- Need to do the product work (could be defined more or less sharp).
 - Probably stuff like: relearn model, do analysis, boring stuff
 - This can also include research (but needs to be tracked as well).
- Tip: Be proactive and create proposals:
 - Come up with an idea how to improve things (new algorithms, new features, ...)
 - Show clearly the business value of this (in terms of dollars!)
 - Help your boss to sell this to the product world

ML Product Development

Do we need Machine Learning?

- Because we can do ML does not mean that we need to do ML! Many people see nowadays ML problems everywhere...
- Do not use ML when you can solve your problem with a manageable set of deterministic rules that you can write down and that would not be too complex to maintain.
- Why? Building, running and operating ML system is very complex, even for simple problems!
- Always start from a concrete business problem and then determine whether it really requires ML.



Start with a simple model

- Once you figured out the business case and made sure that ML can really make a difference, start with a very simple model (logistic regression or simple decision rule) to have a baseline.
- Setup the whole system around that base model
 - Gather training data and setup the data pipelines
 - Clearly define the label (not always so obvious)
 - Decide on how to measure performance (technically and in €)
 - Decide on how to deploy the model and how to monitor it
- Start improving the base models once all other problems are solved.

More Useful Resources

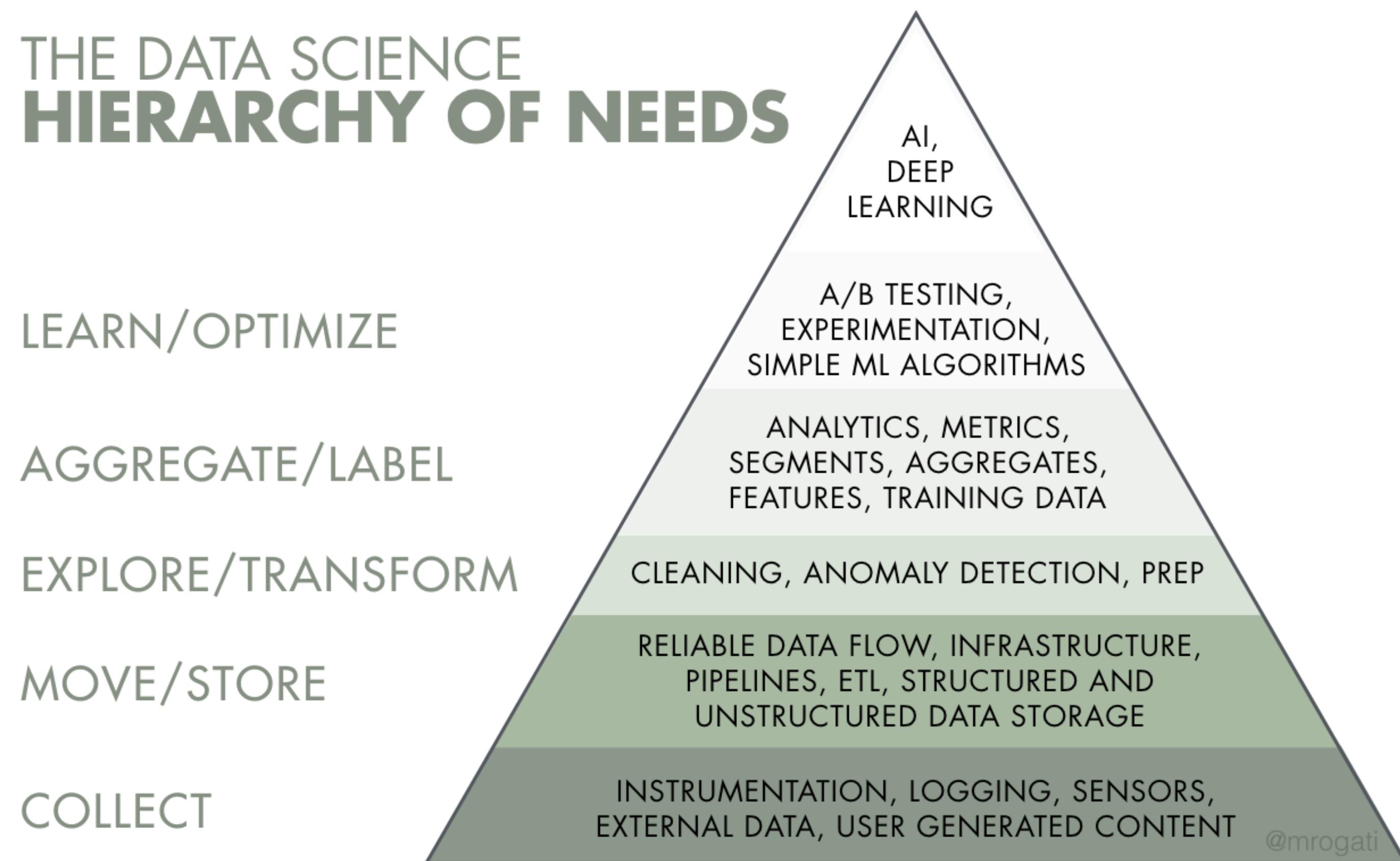
Data Science Manifesto

A set of rules of best practices for Data Science in the wild:

<http://www.datasciencemanifesto.org/>

- Minimal Viable Products over prototypes
- APIs over databases
- Clever use of computation over convenient assumptions
- Dashboards over reports

The AI Hierarchy of Needs



What is your ML test score

A paper that gives hints about running a machine learning system:

```
@inproceedings{45742,  
  title = {What's your ML test score? A rubric for ML production systems},  
  author = {Eric Breck and Shangqin Cai and Eric Nielsen and Michael Salib  
           and D. Sculley},  
  year = {2016}  
}
```

- Test that the distributions of each feature match your expectations.
- Test the effect of model staleness.
- Test the reproducibility of training
- ... and many more.

Useful resources about practical DS

- What is the Team Data Science Process? ([Link](#))
- Free [online course](#) about full stack deep learning:
 - Infrastructure and Tooling
 - Data Management
 - Machine Learning Teams
 - Training and Debugging
 - Testing and Deployment

Final words

Thanks for participating in this class and let's stay in touch!

- If you need help in your future work: I take part in an [industry transfer program](#), which does DS consulting/freelancing for companies.
- If you are looking for students doing a thesis or an intern.
- If you have a research problem and want to have apply for research grants for a PhD student working on it.
- If you happen to be around Karlsruhe and want to catch-up on Data Science topics (or anything else).