Practical Aspects of Data Science

Data Science Retreat - 2019/B1
Patrick Baier

Course Goal

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

➔ **Model Learning (Day 1)**
   Evaluation metrics, imbalanced data problems, probability calibration

➔ **Model Operation (Day 2)**
   Model deployment, missing features, monitoring

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.

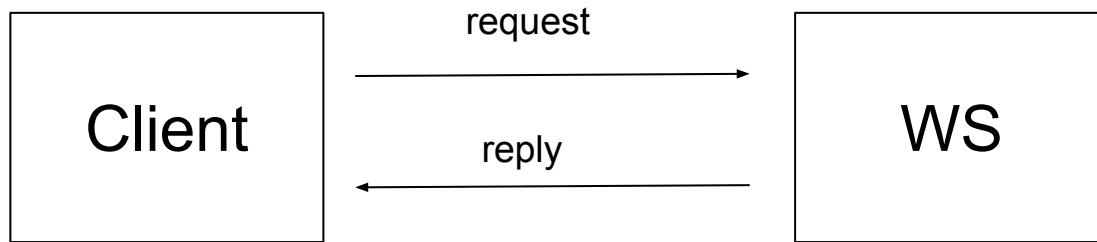→ We have to build a webservice that contains our model:

Input: Feature vector

Output: Return probability

Webservice

Webservice 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.

```
┌──────────┐      request      ┌──────────┐
│          │ ─────────────────▶│          │
│  Client  │                   │    WS    │
│          │◀───────────────── │          │
└──────────┘       reply       └──────────┘
```

Webservice

We provide our model to the outside world through such a service.
i.e. other people send a request in the form:
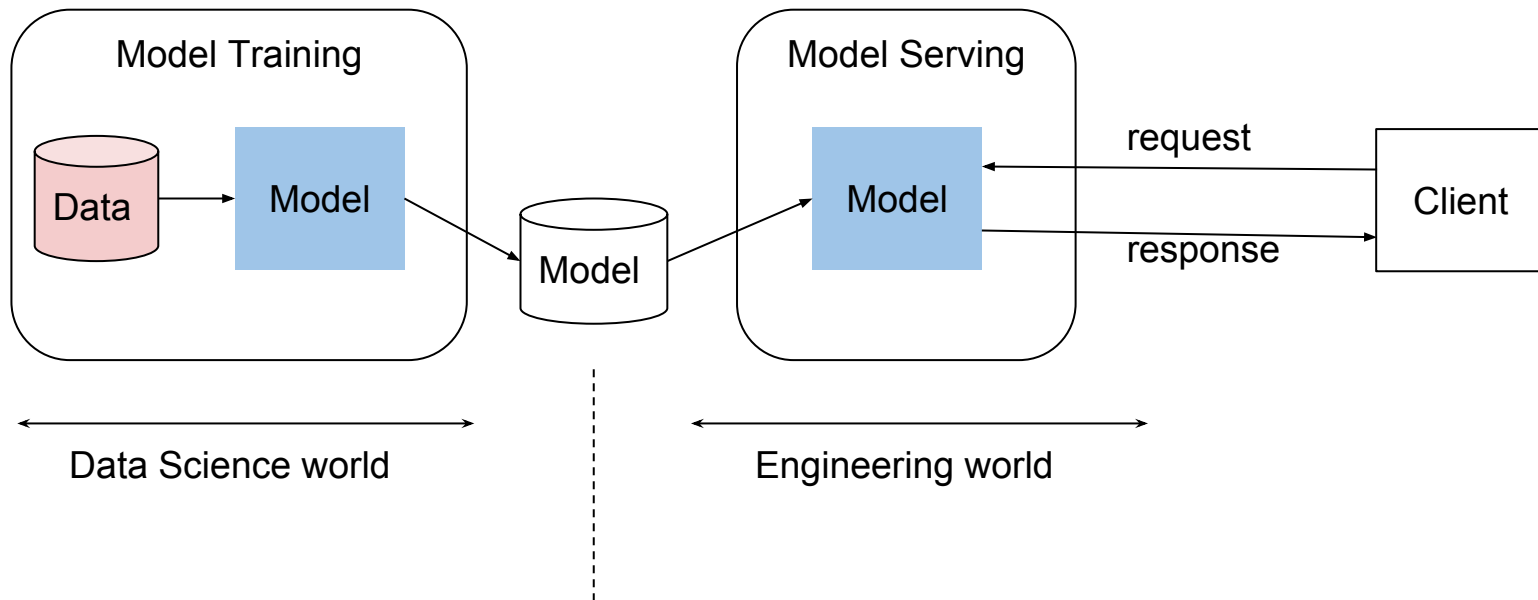
`192.168.1.204:5000/predict`

Your IP   Port of service   Endpoint for predictions

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

Model in Production

Flask Webservice

Let's build a REST service for predictions:

```python
@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
```

Bind method to `predict` endpoint of server

Extract fields from request that was sent by client

Return return probability to client (as json)

Task 1

1. Get the webserver up and running on your machine.
   Execute in a shell: `python server.py`

2. Execute `request.test` to send a requests to your server:
   Execute in a different shell: `source request.test`

For Windows users:
> set FLASK_APP=server.py
> python -m flask run
> 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

- Extend the webserver to handle requests using the ML model.
  - Save your model in the jupyter notebook, example:
    http://scikit-learn.org/stable/modules/model_persistence.html

  - Load the model into your server
  - Predict with the model on the incoming request

- Run the request simulation tool to test your server:
  Execute in a shell: `python3 loadSimulator.py`

Missing Features

Problem

- We often have data points where one or more features are missing/nulls: $<x_1, x_2, \texttt{null}, x_4>$
- We cannot train or predict on only a subset of features

Why are values missing?
- network problems
- timeouts
- optional fields
- unavailability
- ...

What can we do?

Feature is missing in live system:

- Imputation
- Multiple models

Feature is missing in training:

- Drop data point
- Value imputation
- Model imputation
- Mixture model

Live - Imputation

Fill up a feature not available at runtime with:

- median values of training data
- mean value of training data

Continuous feature

- median values of training data
- majority value of training data

Discrete feature

Live - Multi models

Learn mode with reduced features:

Normal model: $f(x_1, x_2, x_3, x_4)$

Reduced models: $f(x_2, x_3, x_4), f(x_1, x_3, x_4),$ ...
                            $f(x_3, x_4), f(x_2, x_4), f(x_2, x_3),$ ...
                            $f(x_1), f(x_2),$ ...

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.

Training - Drop data

dropping incomplete features
→ drastically reduces data

Training - Value imputation
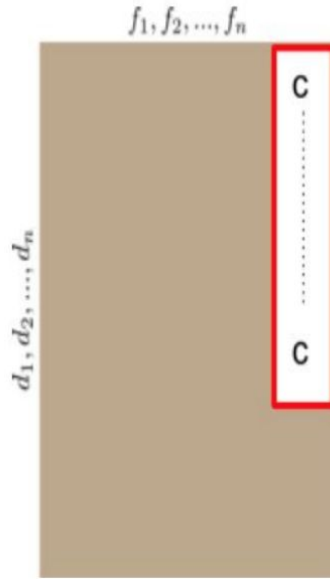
Value imputation:
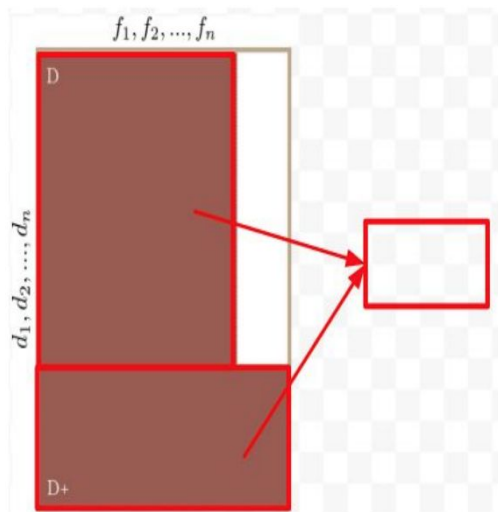
- mean
- median
- value out of definition scope

Training - Model imputation
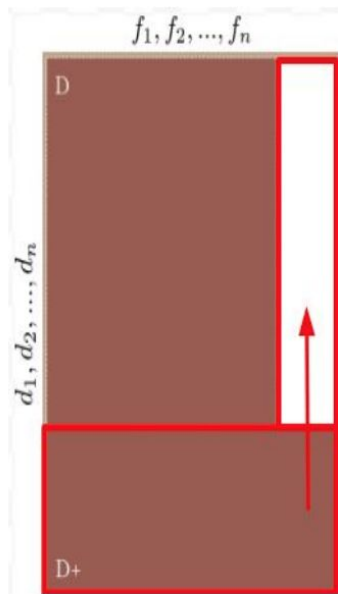
Model imputation:

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

Training - Mixture models

Mixture models:

- 2 submodels:
    1. complete data points
    2. reduced feature set
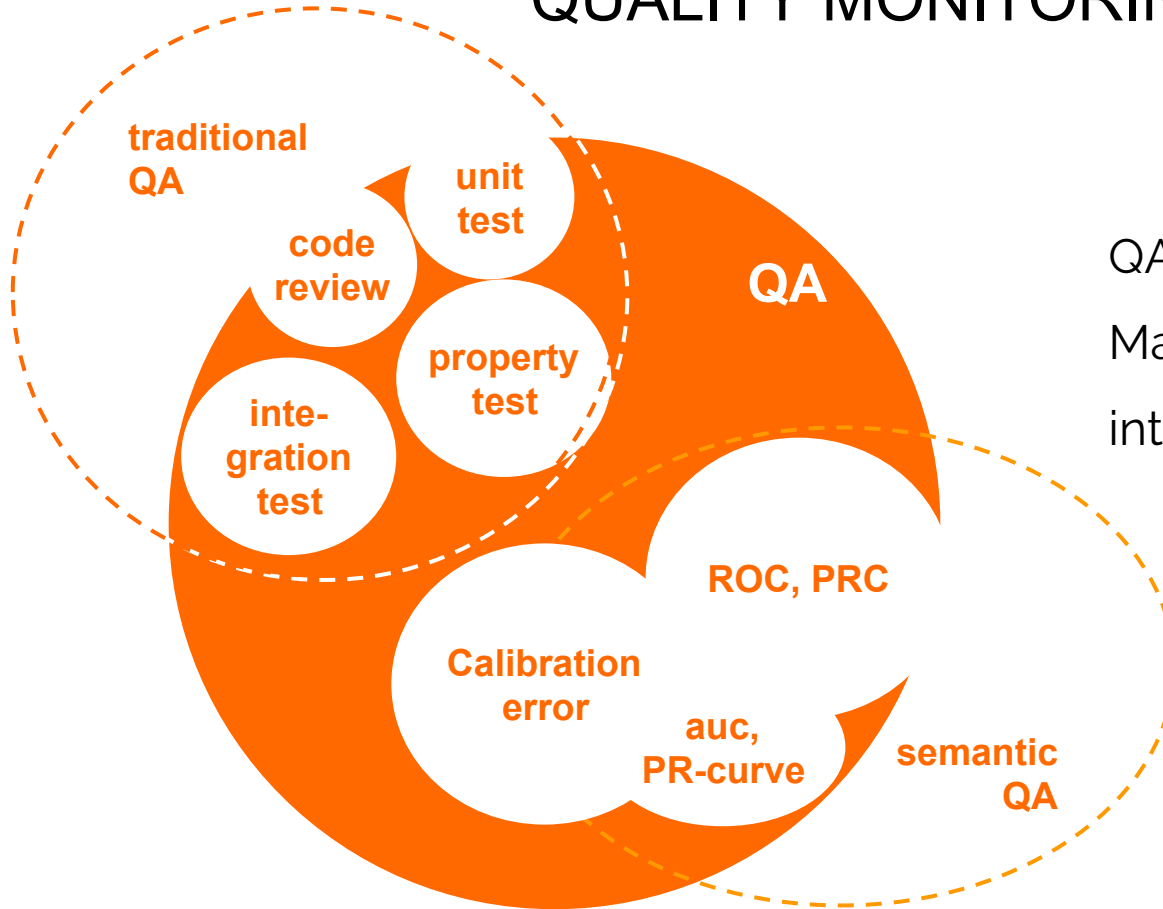- use their output as input for decision model

Task

1. Extend the web server to be able to deal with missing features:
   - Create an imputer on top of the training data.
   - Use this imputer in the flask server.
2. Run the request simulation tool again.

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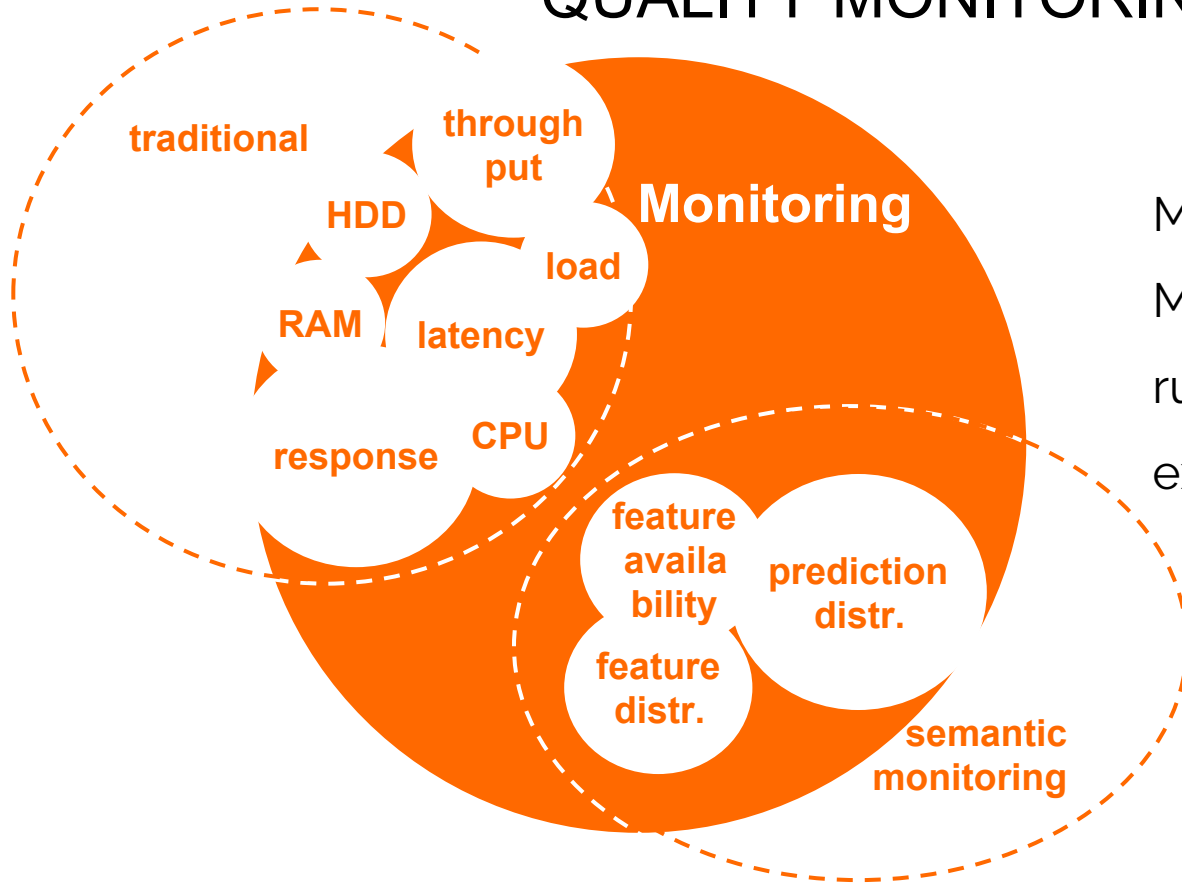2. Do not use the sklearn library

Monitoring

QUALITY MONITORING

QA (Quality assurance):

Make sure that what goes

into production is good

Monitoring

traditional
through put
HDD
load
RAM
latency
response
CPU

feature availability
prediction distr.
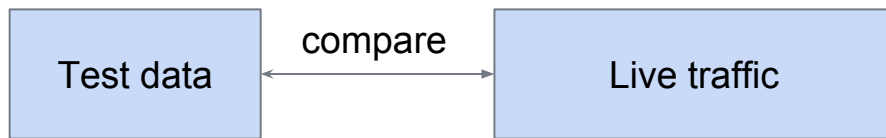feature distr.
semantic monitoring

Monitoring:

Make sure everything that runs live is working as expected

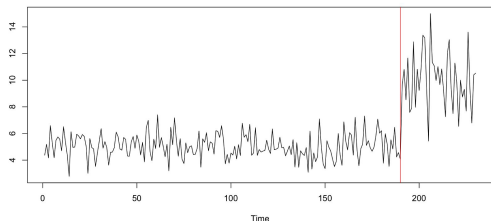# QUALITY MONITORING

There are two kind of model monitoring:

1. Check if model is working before it goes live.

Test data	compare	Live traffic

2. Check if model is working continuously in live.

Monitor model with change point detection
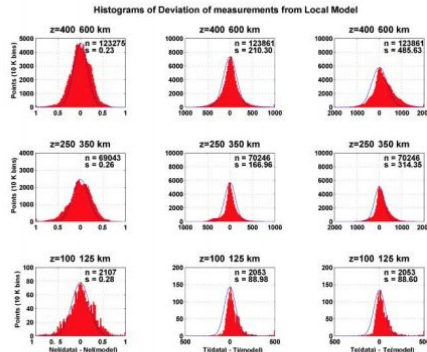
QUALITY MONITORING

1. Check if model is working before it goes live:

   - We simulate the model with one week live traffic.
   - We check against test data:
     - Failing features
     - Feature distribution
     - Probability distribution
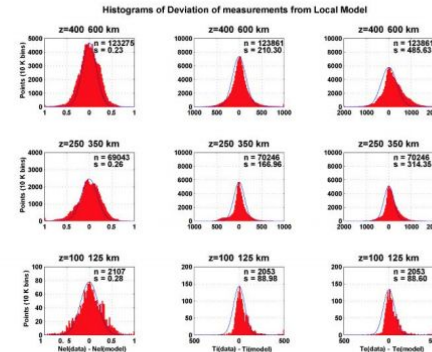   - If all looks similar we go live

QUALITY MONITORING

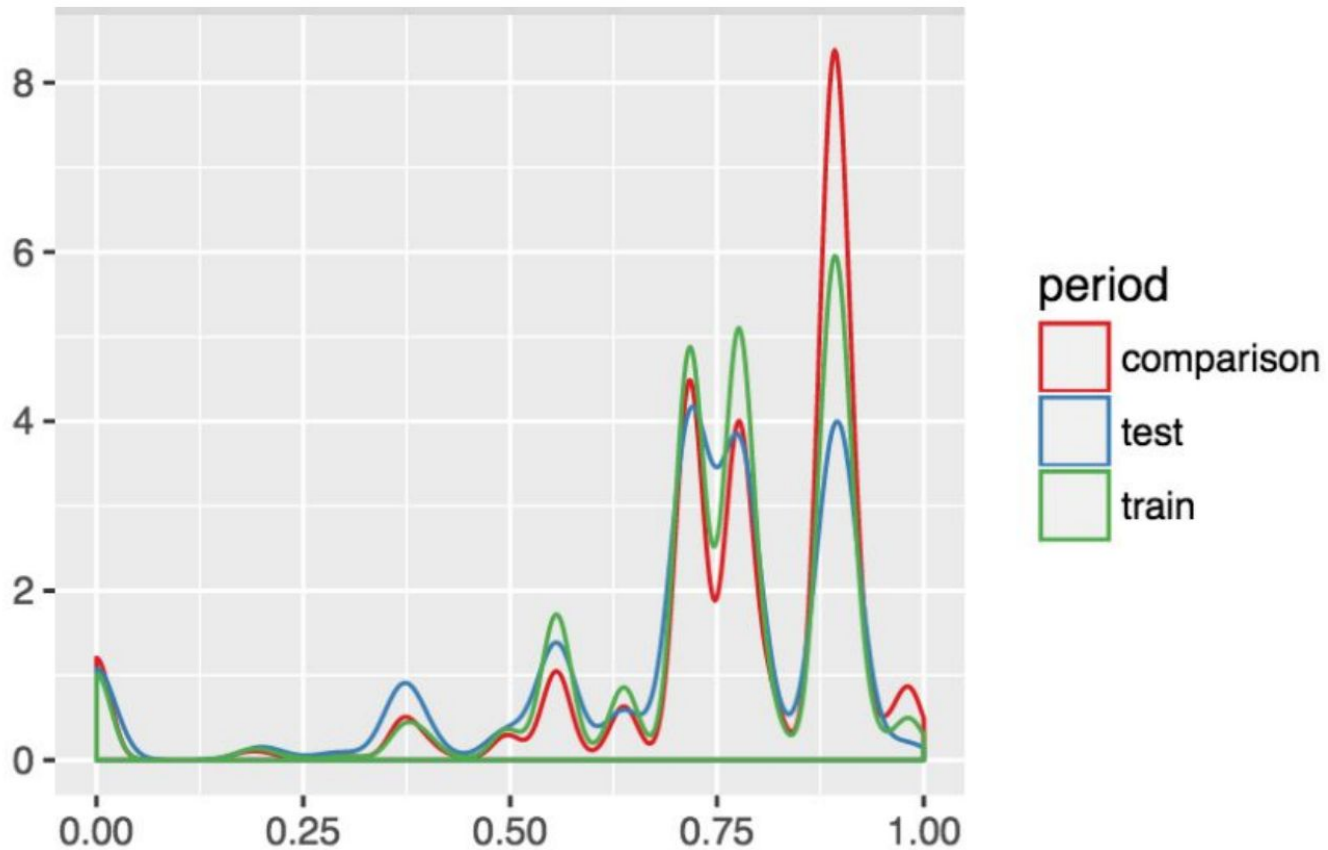Compare feature distributions and output probability:

QUALITY MONITORING

QUALITY MONITORING

Compare distributions with some distance:

feature name	this vs previous	this vs test	previous vs test
feature one	0.000008	0.928806	0.928798
feature two	0.0009117	0.019504	0.020416
feature three	0.1075305	0.316970	0.313337
feature four	0.943896	0.943655	0.045654
...	...	...	...
prediction	6.606939e-02	0.255182	0.277325

QUALITY MONITORING

possible discoveries

technical problems,
seasonalities,
change of behaviour,
deviation from expectations.

Distribution Distance

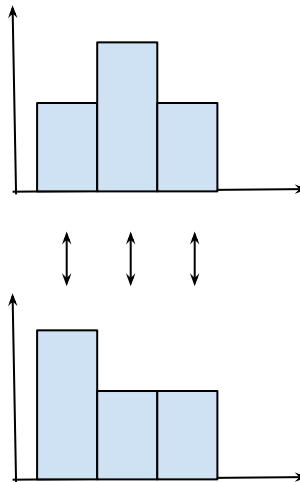How do we measure the distance between two distributions?
Answer: There are statistical test for that!

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

Distribution Distance

We can also use a simple distance measure:

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

Task

Let's build such a system!

1. Integrate a logging mechanism into the flask server:
   Write every request and the predicted probability to a logfile.

2. Test the logging with the loadSimulator. Increase the request speed
   of Line 59: `time.sleep(0.01)`

Task

3. Built on a monitoring system on top of these logs.
    Requirements:.
    -   Give out the share of missing features for every feature.
    -   Compare the distribution of all features after 500 requests using against the test data.

QUALITY MONITORING

2. Check if model is working before it goes live:

    - We simulate the model with one week live traffic.
    - We check against test data:
        - Failing features
        - Feature distribution
        - Probability distribution
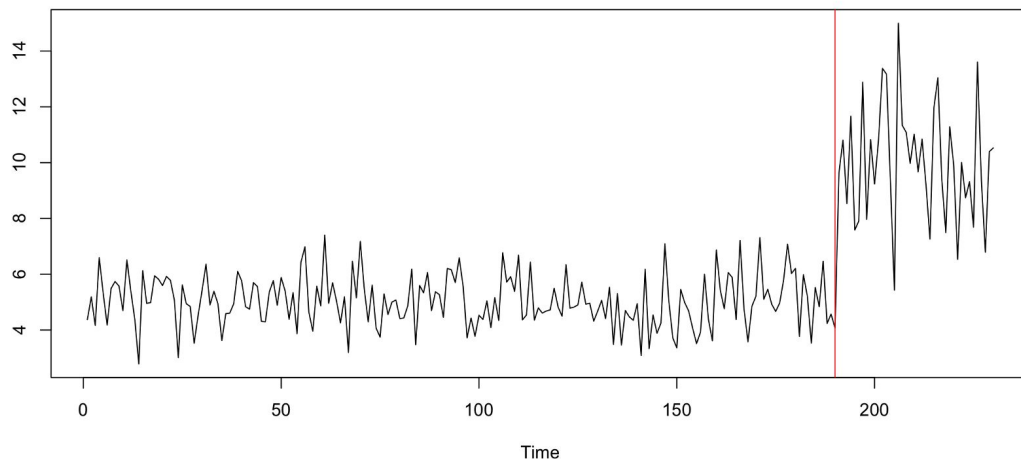   - If all looks similar we go live

QUALITY MONITORING

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

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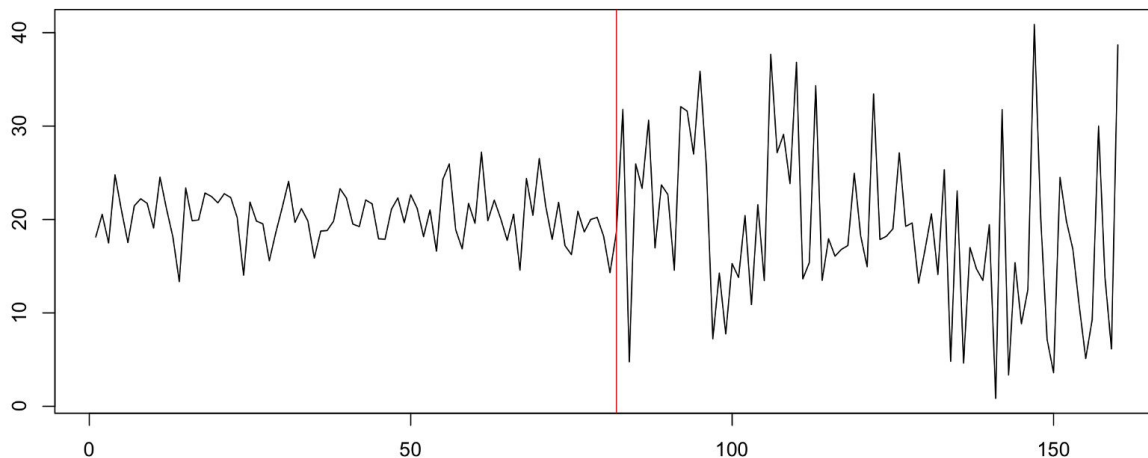

Change in **mean** of time series

QUALITY MONITORING

Idea: Check if a time series changes is 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, Jaxk, et al. "A review and comparison of changepoint detection techniques for climate data." Journal of Applied Meteorology and Climatology 46.6 (2007): 900–915.

Task

4. Implement a change point detector as the algorithm suggests it on the last slide (use the provided amount.log):
   - The algorithm should work in an online-fashion (consuming one data point after another).
   - The sliding window should contain 100 values.
   - The alert should pop up if the mean shifts by more than 5%.

Change point detection

For our case, let's use an rather simple algorithm to tackle this:

```
DetectShift():
```

1. Define a big window w1 and a small window w2
2. Iterate over the series and for every new data point calculate the mean of the windows.
3. Alert if the both deviate by more than epsilon
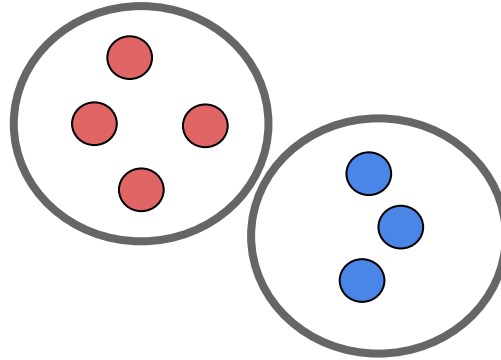
Data Science Organisation

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.
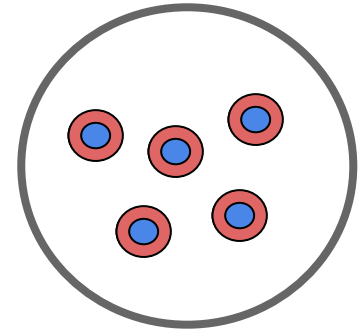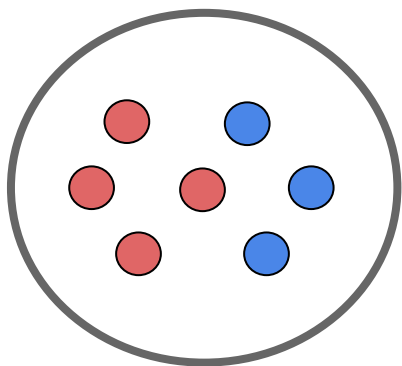
DS Team Structure
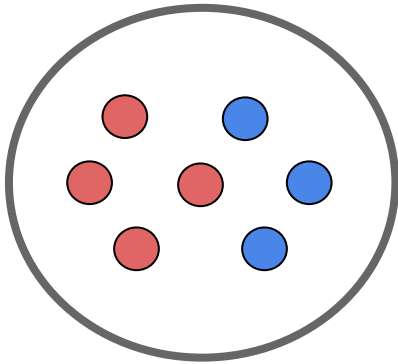
Mixed Team   Separate Team   Hybrid Team

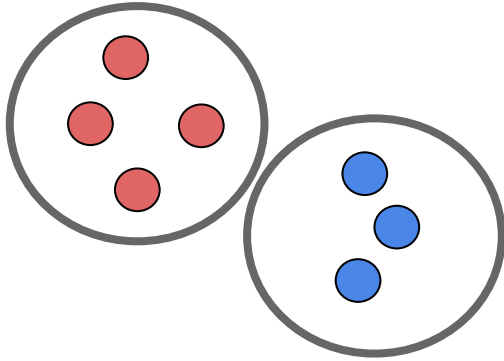Data Scientist
Engineer

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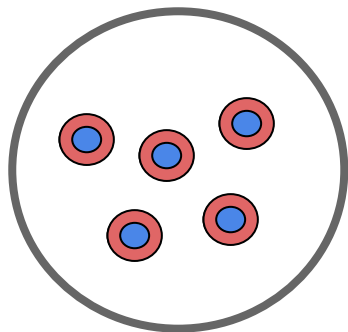
Engineer
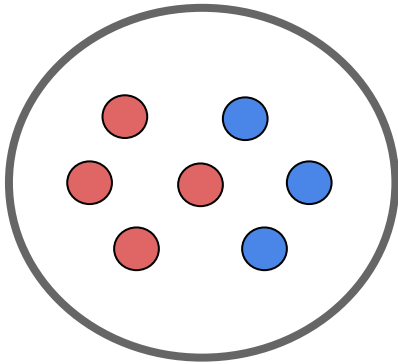
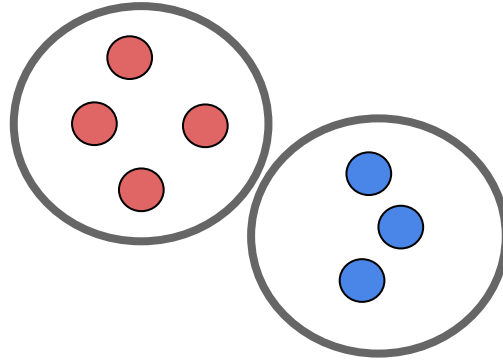- People in this team do both: Engineering and DS.
- Often found in small start-ups.
- Gives you the chance to learn a lot about running a production system (a very valuable asset that will differentiate you from most other DSs).
- But: Make sure that you have still enough DS time as business grows.

DS Team Structure

Mixed Team

Separate Team

Hybrid Team

Data Scientist

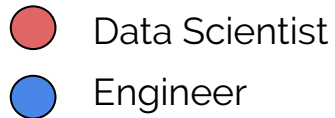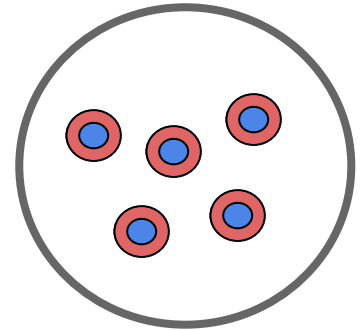Engineer
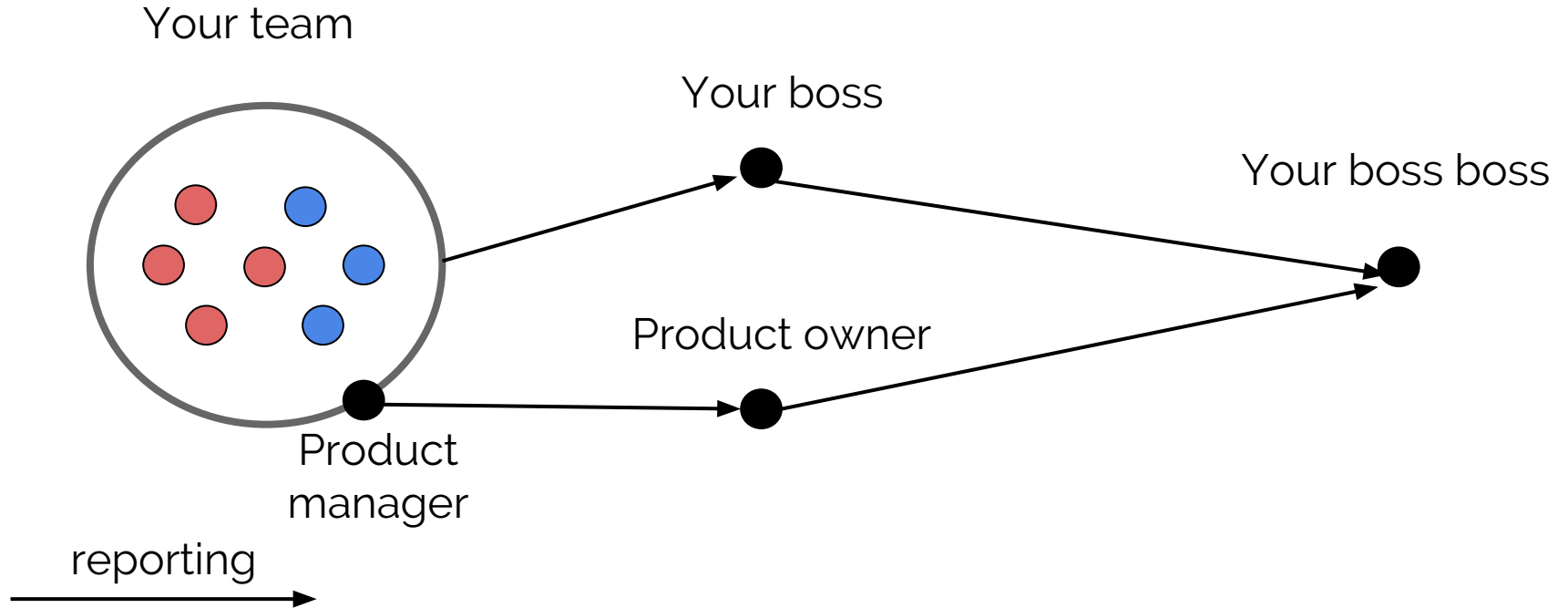
Your Stakeholders

Stakeholders - Product

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.)

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

AI Scientist/Engineer responsibility

- Acquire training data

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

Stakeholders - Bosses

Your boss:

- Coordinates your work and manages you (people management).
- Creates technical vision and talks a lot to product owner.

Your boss boss:

- Manages your boss and the product owner and typically has not much to do with technical details.
- Measures outcome in dollars.

Stakeholders - You

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).
- But: You can (and should) 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

Data Science Manifesto

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

http://www.datasciencemanifesto.org/

The AI Hierarchy of Needs

THE DATA SCIENCE **HIERARCHY OF NEEDS**

AI, DEEP LEARNING

LEARN/OPTIMIZE — A/B TESTING, EXPERIMENTATION, SIMPLE ML ALGORITHMS

AGGREGATE/LABEL — ANALYTICS, METRICS, SEGMENTS, AGGREGATES, FEATURES, TRAINING DATA

EXPLORE/TRANSFORM — CLEANING, ANOMALY DETECTION, PREP

MOVE/STORE — RELIABLE DATA FLOW, INFRASTRUCTURE, PIPELINES, ETL, STRUCTURED AND UNSTRUCTURED DATA STORAGE

COLLECT — INSTRUMENTATION, LOGGING, SENSORS, EXTERNAL DATA, USER GENERATED CONTENT

@mrogati

https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007

What is your ML test score

Let's go through this paper and see what we covered in this workshop:

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

Useful links about practical DS

- What is the Team Data Science Process?
  https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview

- Applying devops to data science
  http://www.hyperbi.co.uk/applying-devops-to-data-science/

- Andrew NG book about DS in practice (http://www.mlyearning.org)