End to end machine learning engineering

Alex Serban

Radboud University, Software Improvement Group, Leiden University
The Netherlands

Who am I?

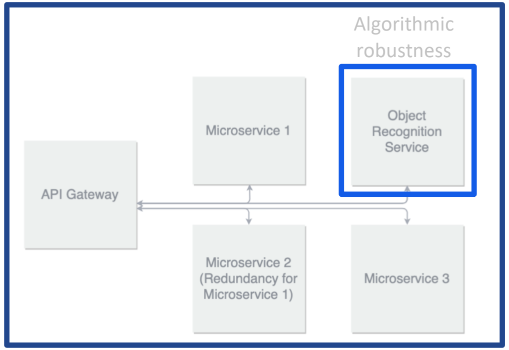
adversarial architecture engineering
examples learning
machine ml pomdp reinforcement
robust safety software
uncertainty

https://se-ml.github.io
Software Engineering for Machine Learning

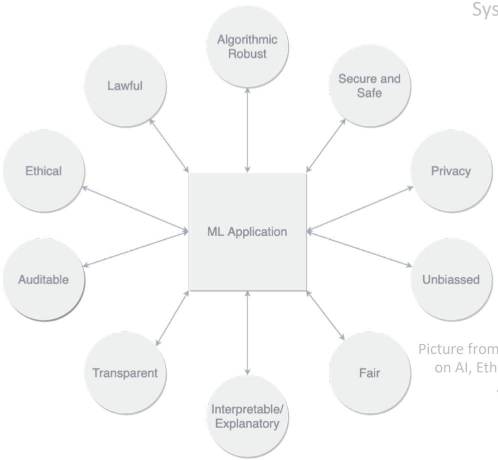
Alex Serban
cs.ru.nl/~aserban

Robustness of autonomous systems

- Robustness has multiple facets, e.g., algorithmic robustness, system or software robustness
- Algorithmic robustness describes the ability of an algorithm to maintain training performance when tested on new and noisy samples
- System robustness describes the ability of a system to cope with errors and erroneous inputs during execution
- When machine learning is used, robustness is broader and includes trustworthy concerns such as fairness, privacy, transparency, etc.



System robustness



Picture from the EU High-Level Expert Group on AI, Ethics Guidelines for Trustworthy Artificial Intelligence

Alex Serban
cs.ru.nl/~aserban

Robustness in the wild

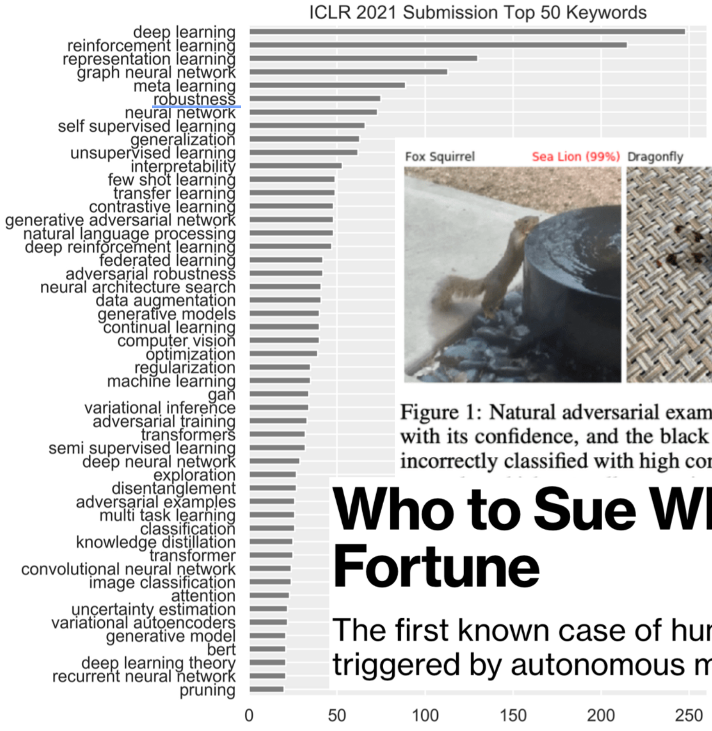
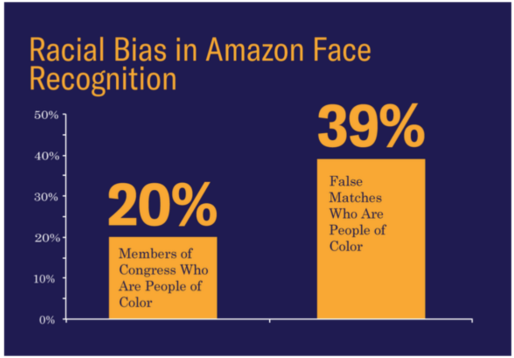


Figure 1: Natural adversarial examples from IMAGENET-A. The red text is a ResNet-50 prediction with its confidence, and the black text is the actual class. Many natural adversarial examples are incorrectly classified with high confidence, despite having no adversarial modifications as they are

Who to Sue When a Robot Loses Your Fortune

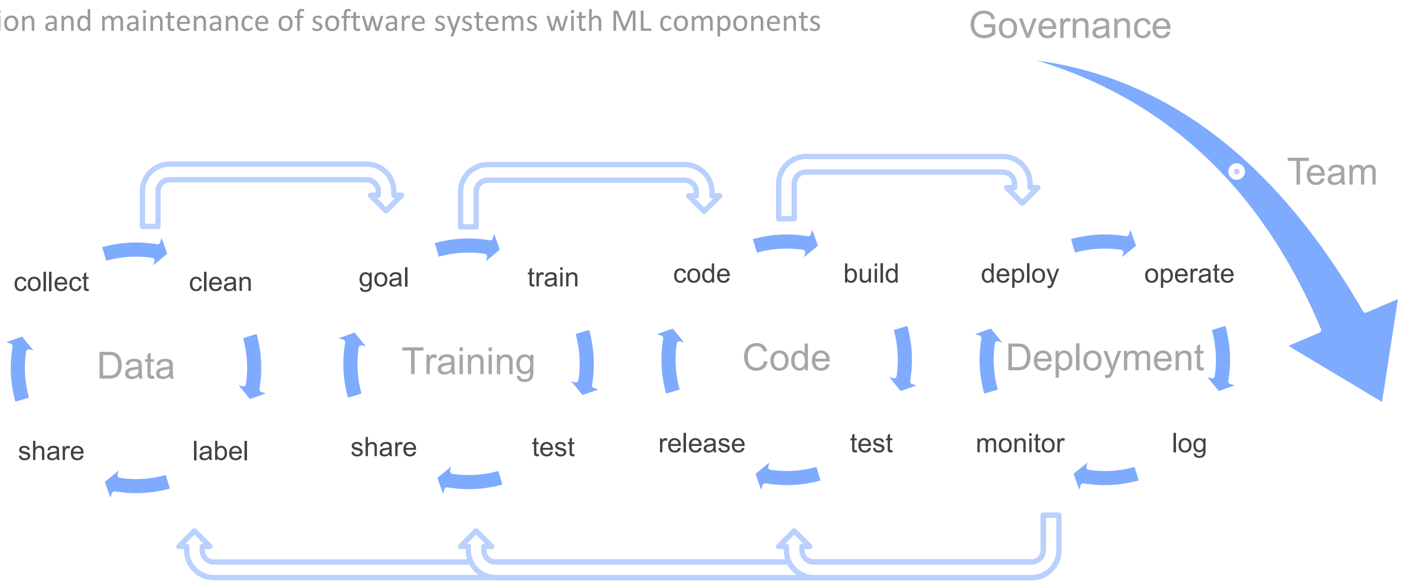
The first known case of humans going to court over investment losses triggered by autonomous machines will test the limits of liability.



Alex Serban
cs.ru.nl/~aserban

End to end machine learning engineering

- The development of engineering principles for the design, development, operation and maintenance of software systems with ML components



Alex Serban
cs.ru.nl/~aserban

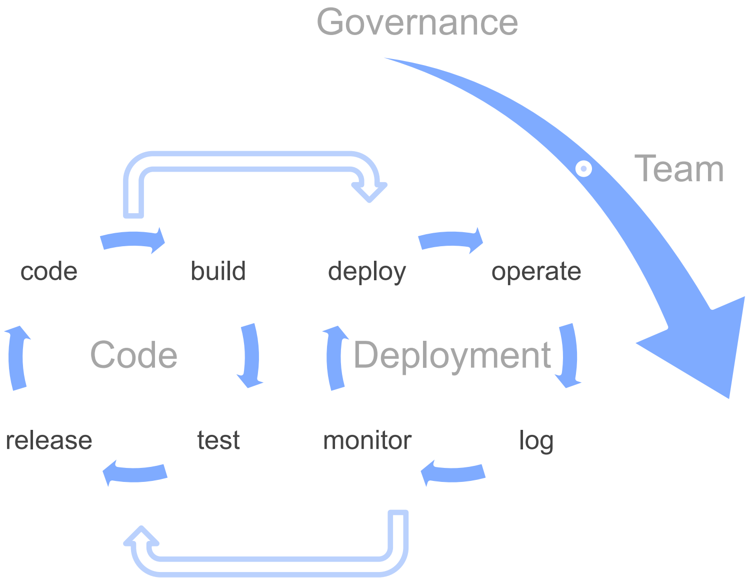
“Traditional” software engineering

- Traditional software engineering tackles challenges related to software design, development and operation

- Such challenges can be classified in functional and non-functional

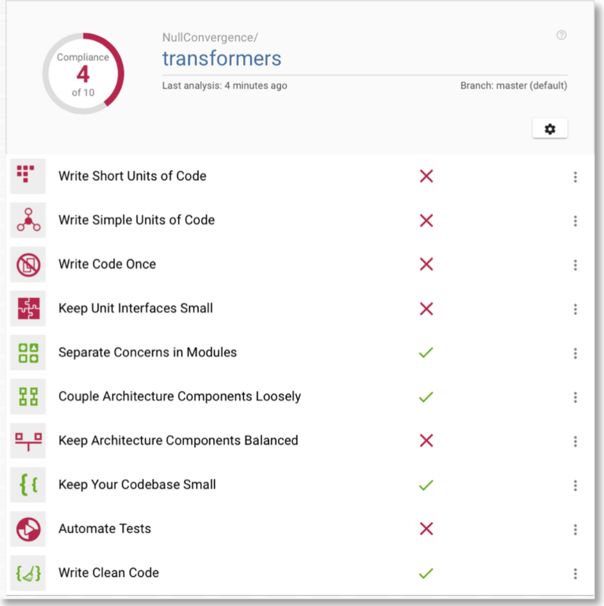
- An example of functional SE challenge is verifying that a system will satisfy its intended functionality (e.g., through testing or formal verification)

- Examples of non-functional SE challenges are maintainability, scalability, usability, etc. (also called “-illities” due to their suffix)

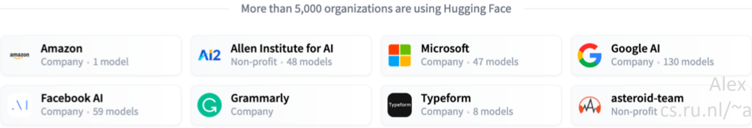


”Traditional” software engineering for ML

- Traditional software engineering practices are also relevant for ML projects
- The tool support for checking traditional practices is mature and openly available (typically free of cost)
- However, in ML systems traditional software engineering practices are not prioritised
- Contributing factors are general unawareness of best practices due to heterogeneous backgrounds
- As research code is cloned and modified, these issues perpetuate

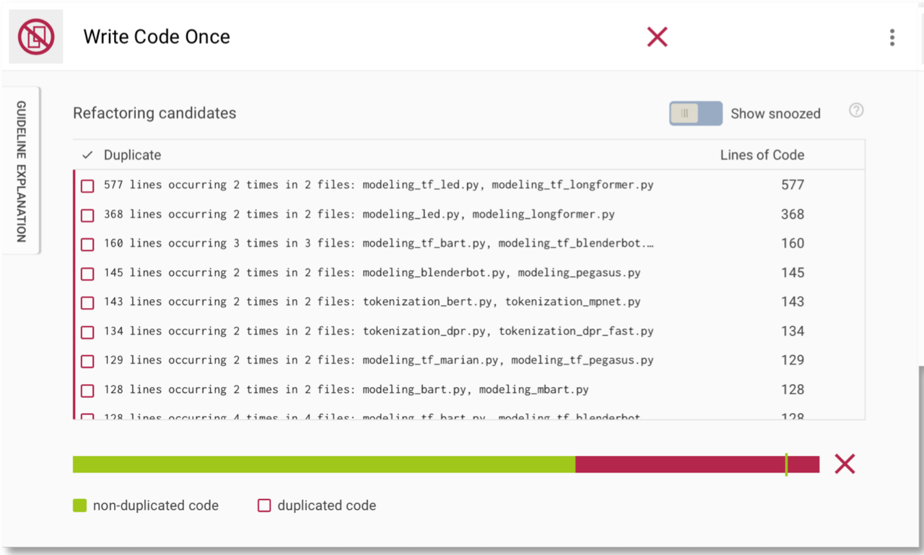


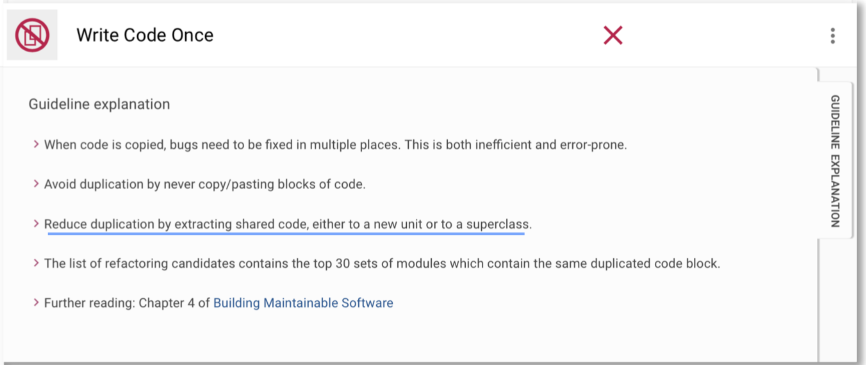
Picture generated by forking the huggingface/transformers repository and running the BetterCodeHub tool



Alex Serban
cs.ru.nl/~aserban

Concrete software engineering for ML

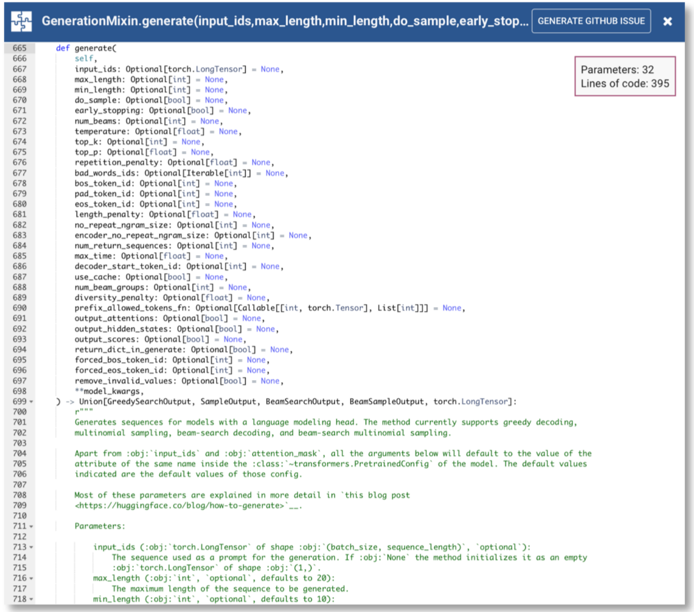




Pictures generated by forking the huggingface/transformers repository and running the BetterCodeHub tool

Benefits of “traditional” software engineering

- Research in software engineering has shown **benefits** of tackling these issue in terms of maintainability, reusability and general effort reduction
- To facilitate adoption of engineering principles by practitioners, they must be **actionable**
- Adopting ”**off-the-shelf**” solution from traditional software engineering in ML should entail similar results
- **Challenge**: Run a static analysis tool on some of your research/framework prototypes and reflect on the outcomes

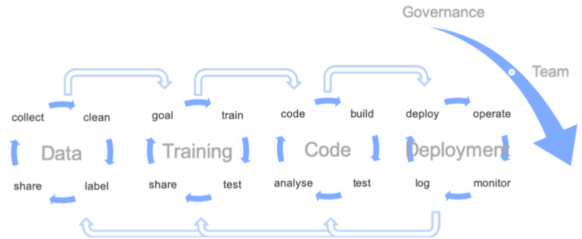


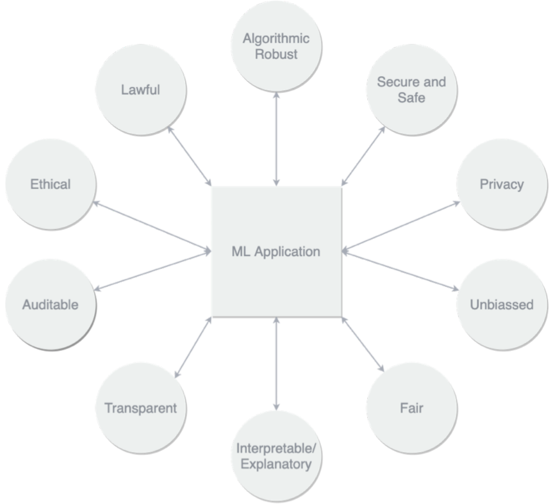
Pictures generated by forking the huggingface/transformers repository and running the BetterCodeHub tool

Alex Serban
cs.ru.nl/~aserban

Machine learning engineering

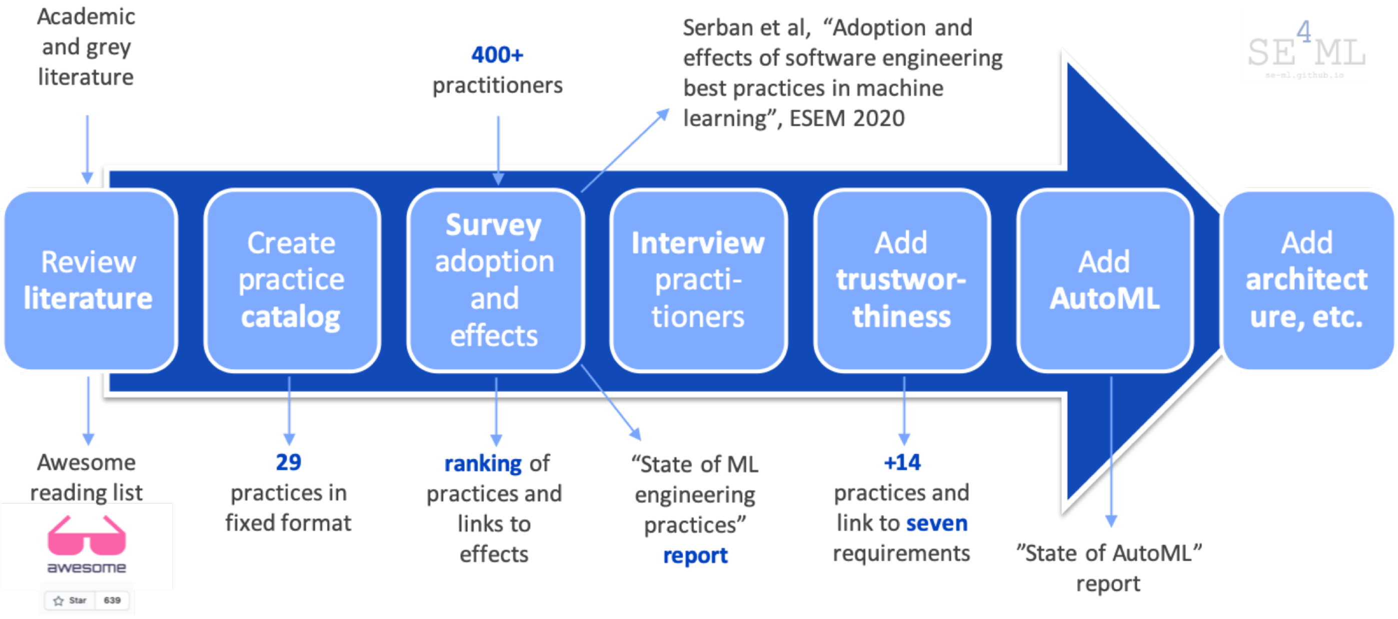
- Machine learning extends traditional software engineering concerns along the following dimensions:
  - *Data-driven behaviour*: the development effort for data management is high, and many challenges regarding e.g., bias, fairness, privacy arise
  - *Inherent uncertainty*: the behavior is probabilistic (not deterministic) which raises challenges regarding testing and error comprehension
  - *Rapid experimentation:* the development process is experiment based, with short , parallel iterations





Alex Serban
cs.ru.nl/~aserban

Machine learning engineering practices

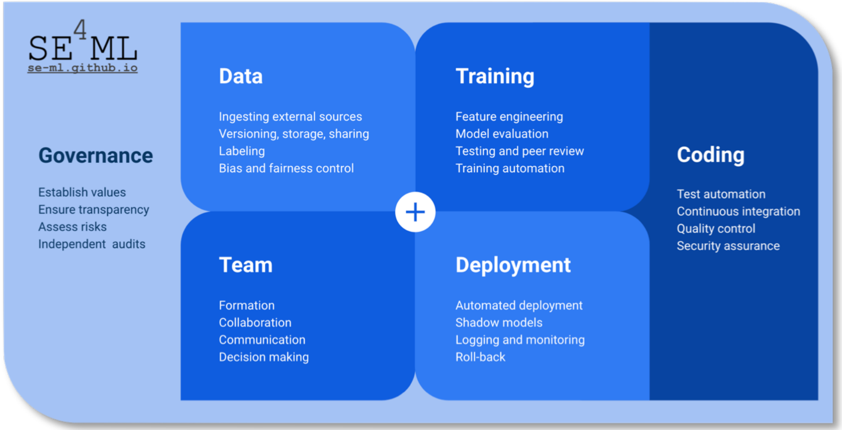


https://github.com/SE-ML/awesome-seml

https://se-ml.github.io/practices

Online catalogue of ML engineering practices

- Originally 29 practices, now grown to 45
- Grouped into 6 categories
- Contains
  - Intent
  - Motivation
  - Applicability
  - Description
  - Adoption
  - Related practices
  - References



https://se-ml.github.io

Ranked on difficulty

Difficulty Basic | Difficulty Advanced

Relation to effects

Effect Quality | Effect Traceability

Relation to EU trustworthy AI

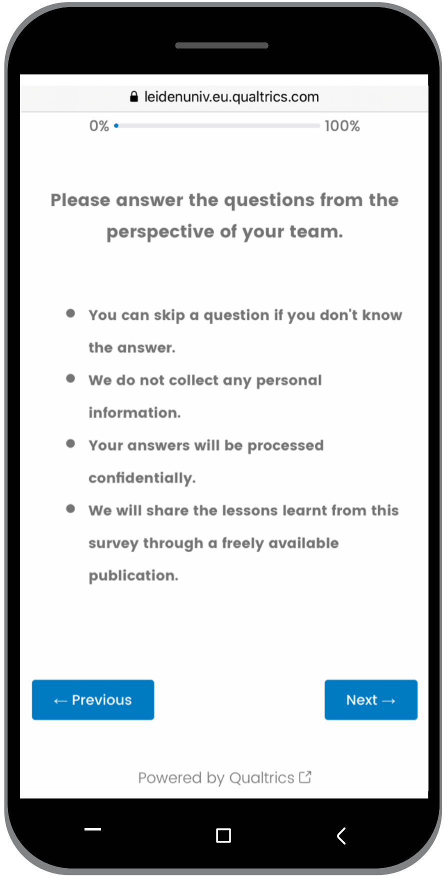
EU Human Agency | EU Privacy

Measuring practice adoption

Survey among teams building software with ML components

Questions:

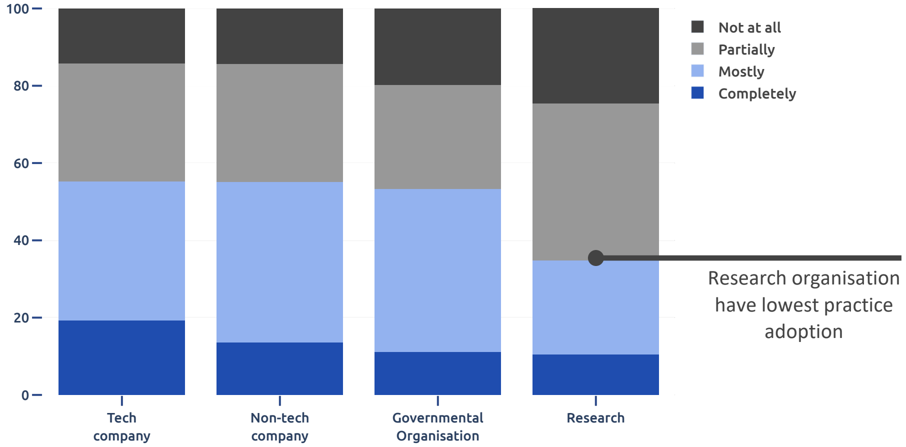
- **General**
  ex. Team size, team experience, country, kind of organization, type of data, tools used.
- **Practices**
  ex. ”Our process for deploying our ML model is fully automated.”
- **Effects**
  ex. “We are able to easily and precisely reproduce past behavior of our models and applications.”



Tech companies lead practice adoption

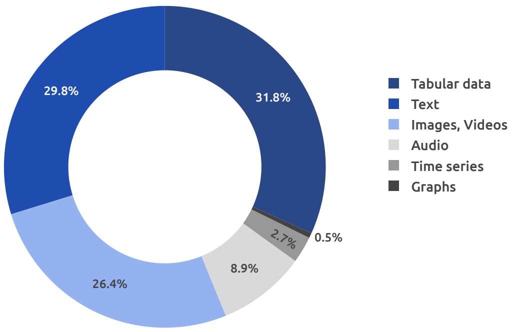
The adoption of best practices by tech companies is higher than by non-tech companies, governmental organizations, and research labs.

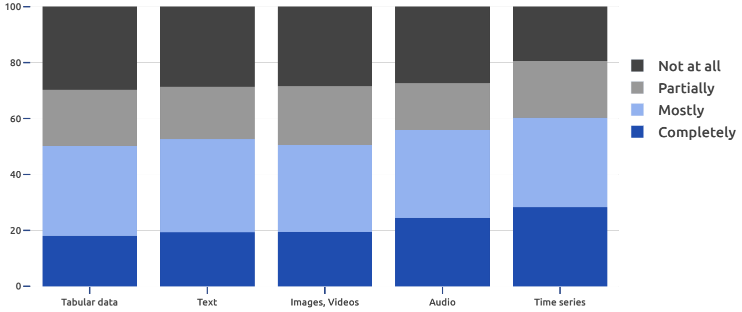
Type of organisation



Practice adoption by data type

The adoption of practices is largely independent of the data type used





Most adopted practices

Practices related to **measurement** and **versioning** are widely adopted.

The top 4 adopted practices are all related to **model training.**

Top 5

1. Capture the training objective in a metric that is easy to measure and understand
2. Share a clearly defined training objective within the team
3. Use versioning for data, model, configurations and training scripts
4. Continuously measure model quality and performance
5. Write reusable scripts for data cleaning and merging

Alex Serban
cs.ru.nl/~aserban

Least adopted practices

The two most neglected practices are related to **feature management**.

Outside research, **Automated ML** through automated optimisation of hyper-parameters and model selection, is not (yet) widely applied.

Bottom 5

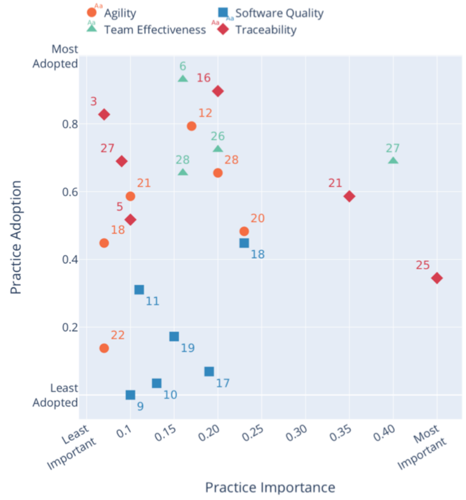
1. Assign an owner to each feature and document its rationale
2. Actively remove or archive features that are not used
3. Run automated regression tests
4. Automate hyper-parameter optimisation and model selection
5. Enable shadow deployment

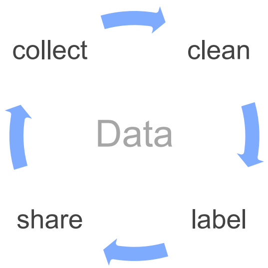
Alex Serban
cs.ru.nl/~aserban

Measuring effects of practice adoption

- For four effects, we hypothesized a relation with specific sets of practices
- Linear regression – confirmed hypothesis
- Random forest – demonstrate non-linear relation
- Importance of each practice using Shapley values – some important practices for the effects have low adoption

Effects	Description
Agility	The team can quickly experiment with new data and algorithms, and quickly assess and deploy new models
Software Quality	The software produced is of high quality (technical and functional)
Team Effectiveness	Experts with different skill sets (e.g., data science, software development, operations) collaborate efficiently
Traceability	Outcomes of production models can easily be traced back to model configuration and input data





ML engineering practices for research

Write Reusable Scripts for Data Cleaning and Merging

March, 2021 • Alex Serban, Koen van der Blom, Joost Visser

4 / 45 • Data • Difficulty Basic • Effect Traceability

Intent

Avoid untidy data wrangling scripts, reuse code and increase reproducibility.

Motivation

Data cleaning and merging are exploratory processes and tend to lack structure. Many times these processes involve manual steps, or poorly structured code which can not be reused later. Needless to mention such code can not be integrated in a processing pipeline.

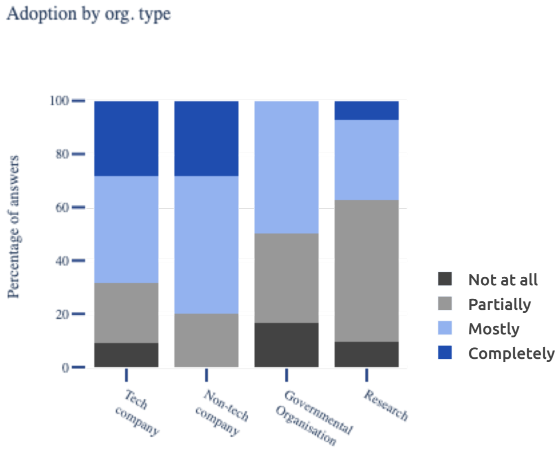
Applicability

Reusable data cleaning scripts should be written for any ML application that does not use raw or standard data sets.

Description

Most of the time, training machine learning models is preceded by an exploratory phase, in which non-structured code is written, or manual steps are performed in order to get the data in the right format, merge several data sources, etc. Especially when using notebooks, there is a tendency to write ad-hoc data processing scripts, which depend on variables already stored in memory when running previous cells.

Before moving to the training phase, it is important to convert this code into reusable scripts and move it into methods which can be called and *tested* individually. This will enable code reuse and ease integration into processing pipelines.



ML engineering practices for research

Share Status and Outcomes of Experiments Within the Team

March, 2021 • Alex Serban, Koen van der Blom, Joost Visser

← 23 / 45 • Training • Difficulty Basic →

Intent

Facilitate knowledge transfer, peer review and model assessment.

Motivation

Team members have different ways of managing and logging experiment related data. Adopting a common way to log experiment data and share it within the team enables members to collectively monitor and assess training outcomes.

Applicability

Experiment tracking and sharing should be used for any training experiment.

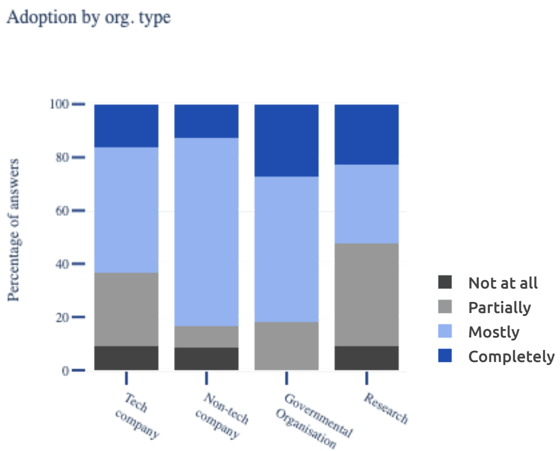
Description

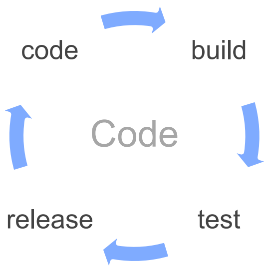
Although different team members have their own style of managing experiments and tracing their outcomes, it is recommended to adopt a common way of logging data; that is understood and accessible to all team members.

Sharing the outcomes within the team has several benefits for peer review, knowledge transfer and model assessment.

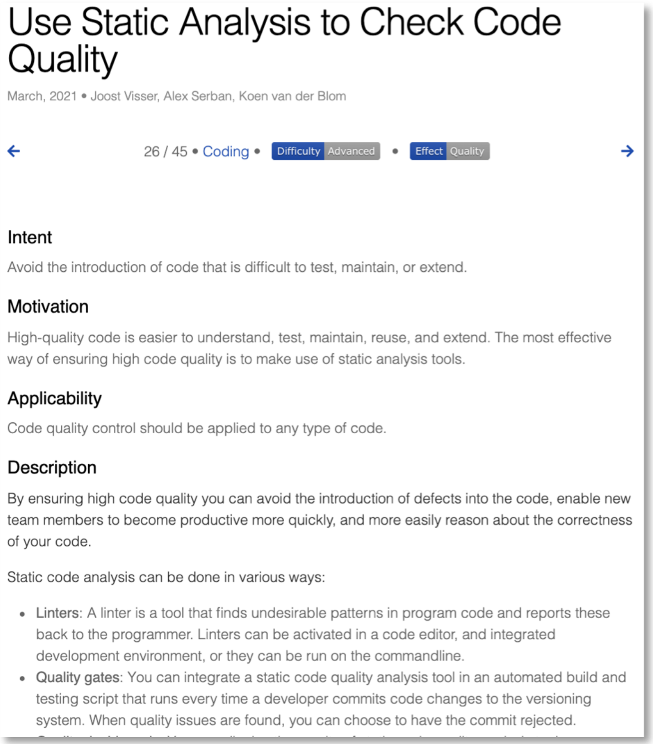
Several collaborative tools enable central logging of experimental results.

Whenever possible, it is recommended to use one of the tools available internally or externally (e.g. Sacred or W&B).





ML engineering practices for research



Use Static Analysis to Check Code Quality

March, 2021 • Joost Visser, Alex Serban, Koen van der Blom

26 / 45 • Coding • Difficulty Advanced • Effect Quality

Intent

Avoid the introduction of code that is difficult to test, maintain, or extend.

Motivation

High-quality code is easier to understand, test, maintain, reuse, and extend. The most effective way of ensuring high code quality is to make use of static analysis tools.

Applicability

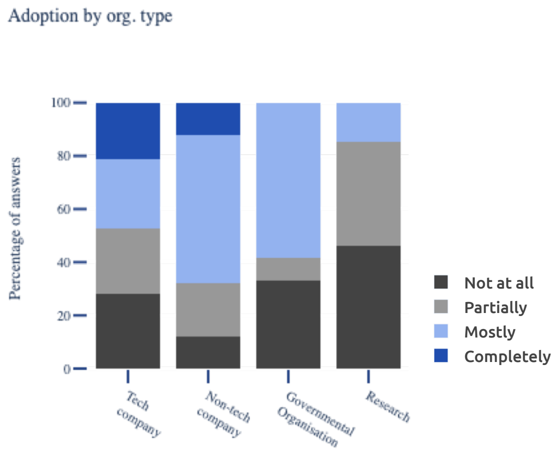
Code quality control should be applied to any type of code.

Description

By ensuring high code quality you can avoid the introduction of defects into the code, enable new team members to become productive more quickly, and more easily reason about the correctness of your code.

Static code analysis can be done in various ways:

- Linters: A linter is a tool that finds undesirable patterns in program code and reports these back to the programmer. Linters can be activated in a code editor, and integrated development environment, or they can be run on the commandline.
- Quality gates: You can integrate a static code quality analysis tool in an automated build and testing script that runs every time a developer commits code changes to the versioning system. When quality issues are found, you can choose to have the commit rejected.



Alex Serban
cs.ru.nl/~aserban

Team

ML engineering practices for research

Use A Collaborative Development Platform

March, 2021 • Joost Visser, Alex Serban, Koen van der Blom

35 / 45 • Team • Difficulty Basic • Effect Effectiveness

Intent

By making consistent use of a collaborative development platform teams can work together more effectively.

Motivation

Collaborative development platforms provide easy access to data, code, information, and tools. They also help teams to keep each other informed, make and record decisions, and work together asynchronously or remotely.

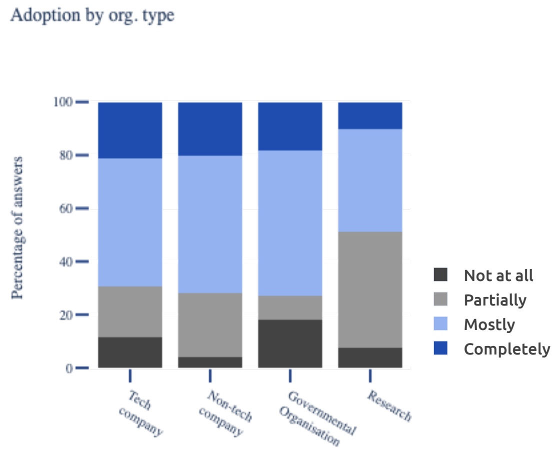
Description

Broadly used collaborative development environments include GitHub, GitLab, BitBucket, and Azure DevOps Server.

Some collaborative development environments are offered as cloud services, others may be installed on-premises, or both. Commonly offered capabilities include:

- Version control
- Issue and progress tracking
- Search, notifications, discussion
- Continuous integration
- A range of developer tools as (third-party) plugins

Collaborative development environments have been developed for, and gained wide-spread adoption by, "traditional" software development teams.



Learn more

Reading list

We reviewed scientific and popular literature to identify recommended practices. Check out our Awesome List with relevant literature.

https://github.com/SE-ML/awesome-seml

Catalogue

The best practices that we identified are describe in more detail in our Catalogue of ML Engineering Best Practices.

https://se-ml.github.io/practices/

Papers

Full details of the methodology behind our survey are described in our scientific articles.

https://se-ml.github.io/publications/

se-ml.github.io

Visit our project website for more details, to take the survey yourself, and to stay up-to-date with our latest results.