School of Electronic Engineering and Computer Science
Queen Mary University of London

ECS7020P Principles of Machine Learning Deployment

Dr Jesús Requena Carrión

7 Dec 2021

Agenda

Machine learning models as products

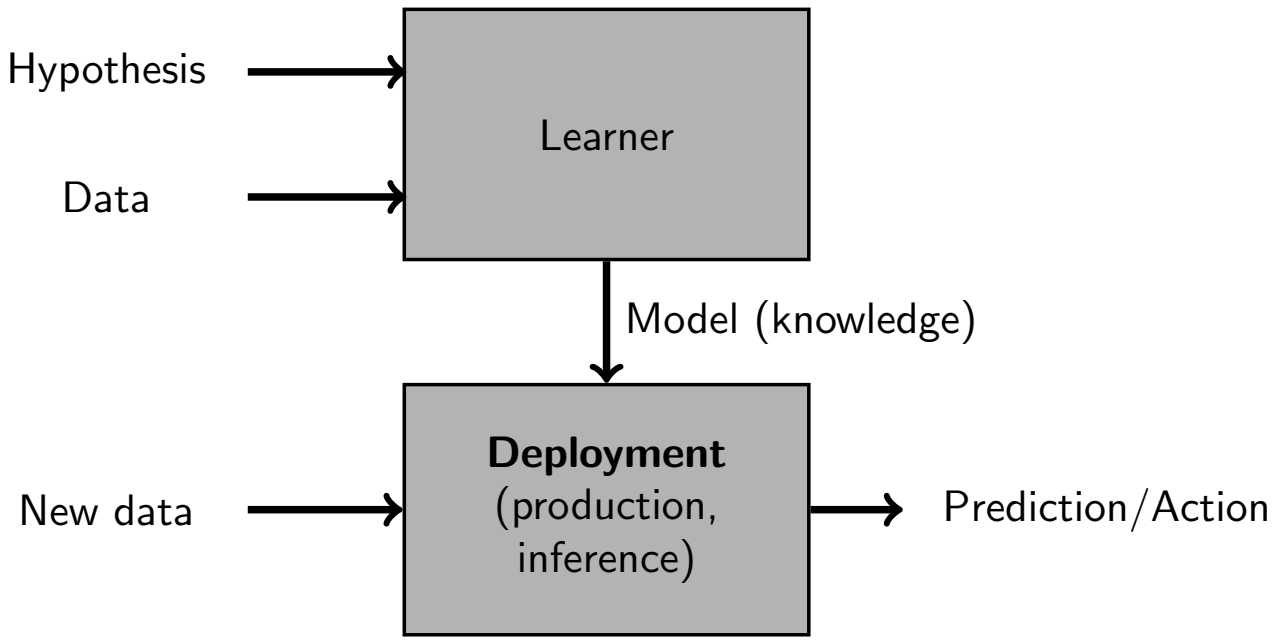
Deployment options

Machine learning system design

ML project management

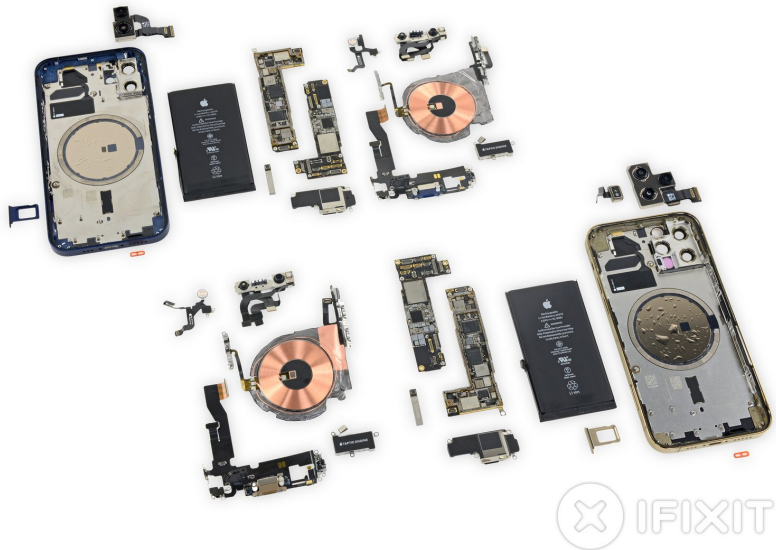
Building and deployment

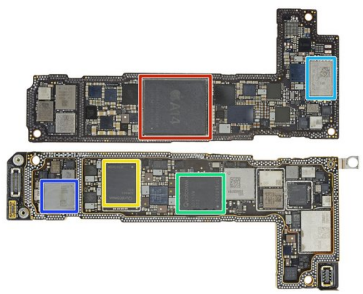
In machine learning we use data to build **models**. Models can be seen as **technological products** that will eventually be deployed to deliver **value**.



A **machine learning system** consists of the data, software, hardware and other infrastructure needed to deploy a machine learning model.

Technological products: Non-ML





https://www.ifixit.com/Teardown/iPhone+12+and+12+Pro+Teardown/137669

iPhone 12 Pro components:

- Apple/Cirrus Logic 338S00565 audio codec
- Qualcomm SDR865 5G and LTE transceiver

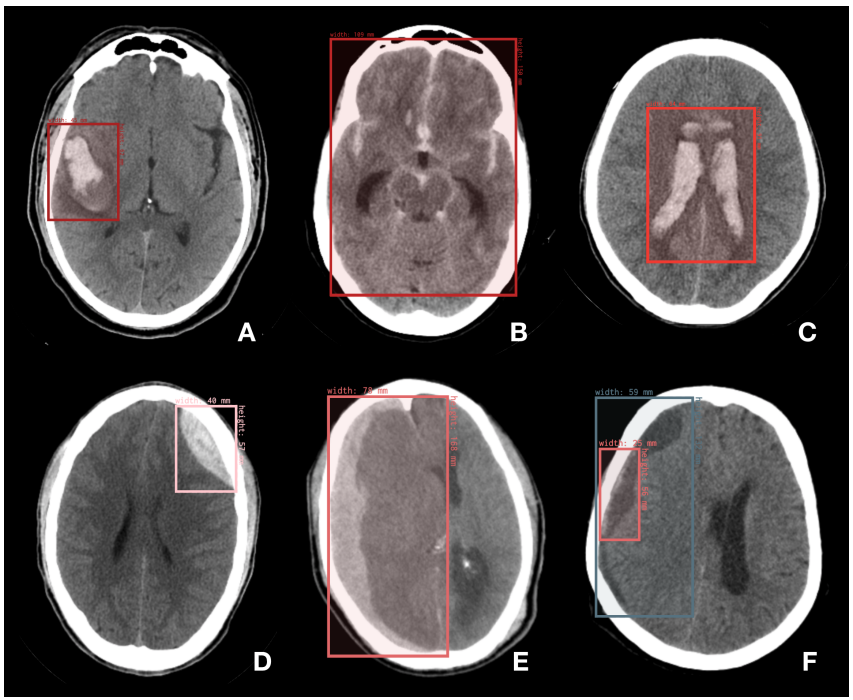
Technological products: ML

https://netflixtechblog.com/artwork-personalization-c589f074ad76

Netflix streaming service:

- Personalisation: one product, one customer
- Concept drift: population changes
- Services always on
- Big: 140M subscribers, 16k titles, 190 countries, several 100K interactions per minute, 90 s window to help find a show

Technological products: ML



Reis, Eduardo Pontes, et al. "Brain Hemorrhage Extended (BHX): Bounding box extrapolation from thick to thin slice CT images"(version 1.1). PhysioNet (2020).

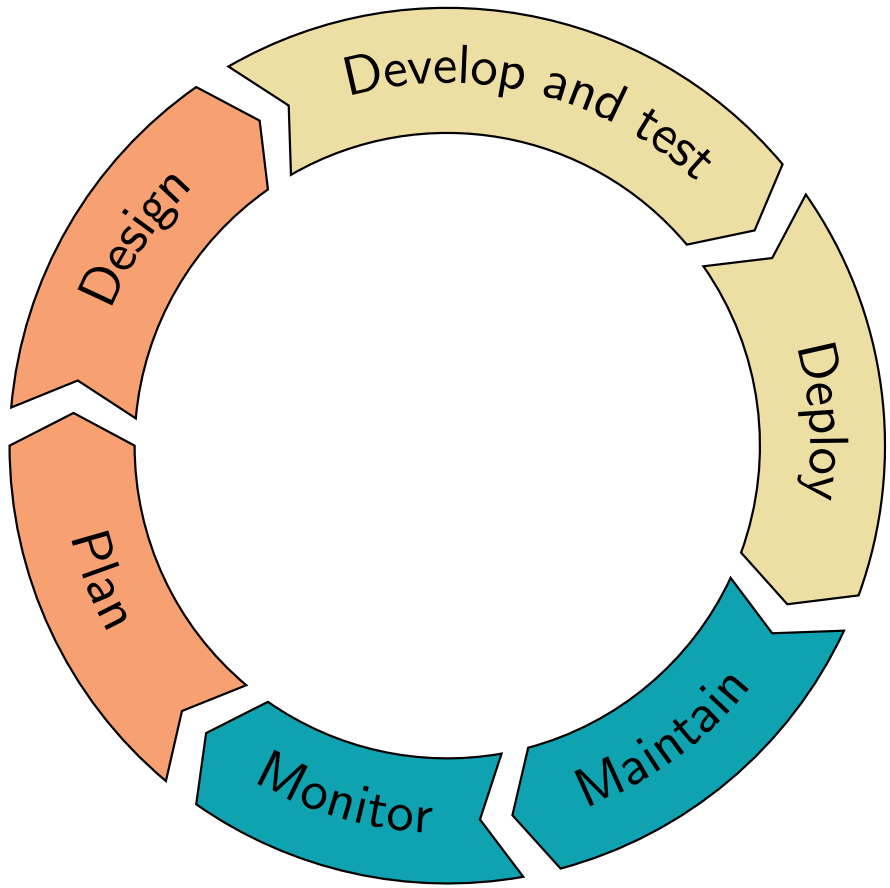
https://doi.org/10.13026/9cft-hg92

Medical image analysis:

- Health Research Authority Approval, Ethics Committee approval
- Data management compliance (GDPR, Data Protection Act)
- Slow data collection and need for skilled annotators

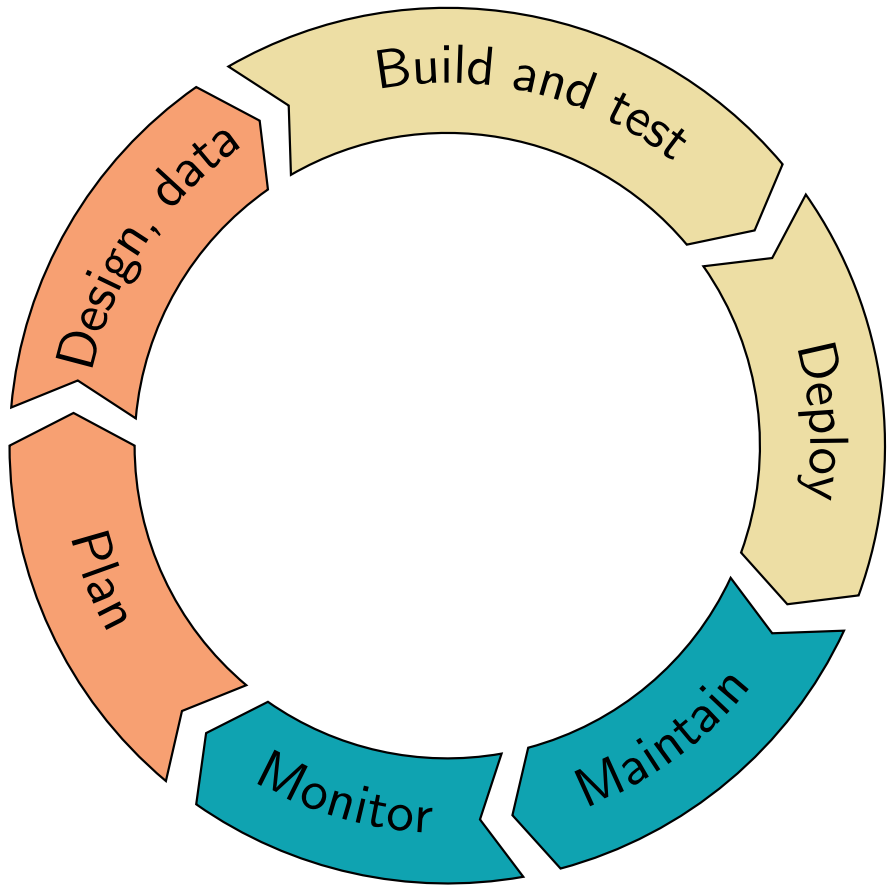
Lifecycle of a typical product

The lifecycle of a product is a business strategic tool that identifies the activities necessary to successfully build and deploy a product.



Lifecycle of a machine learning model

We can use the standard notion of product lifecycle to **manage** models and characterise them using business and management metrics.



Machine learning systems as computing products

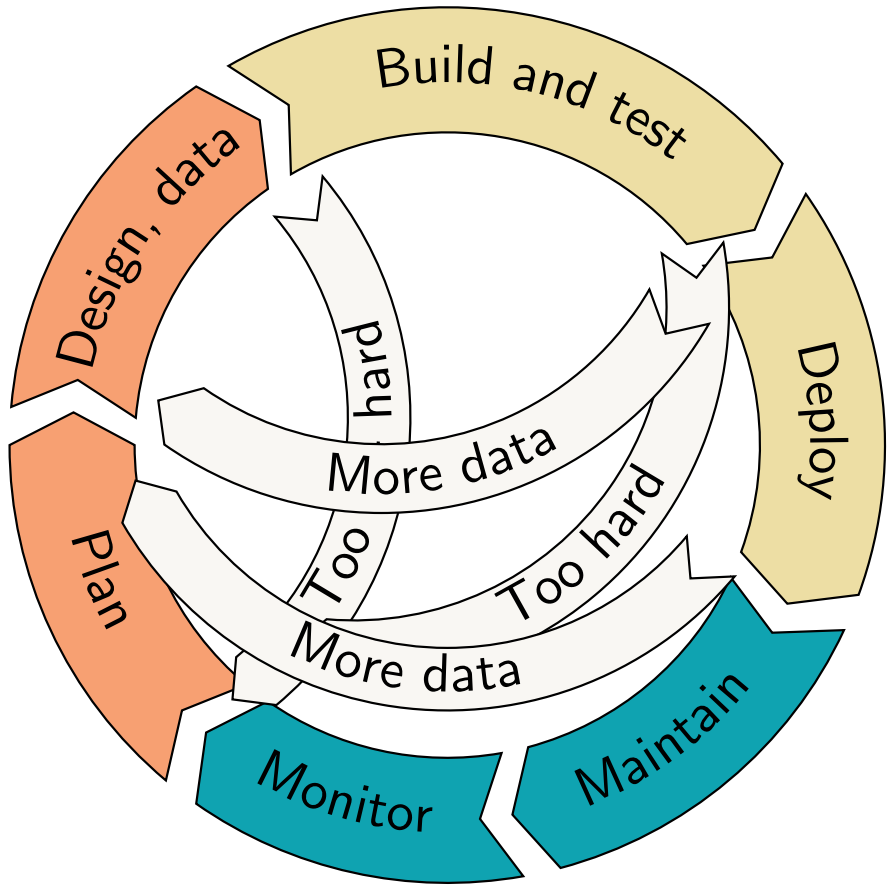
Machine learning systems are **computing products**, therefore we will see **software design concepts** (modularity, maintainability, compatibility, reliability, reusability, versioning...) being used in ML projects.

What makes a machine learning systems different?

- We use **data** to build models. Data representativity, quality and quantity are essential and new data can be used to update models.
- **Optimisation** and **hypterparameter** selection improves models.
- **Testing** involves software performance metrics, business metrics and machine learning metrics, sometimes conflicting.
- Building and deployment environments can be different, which can affect **reproducibility** and **integrability**.

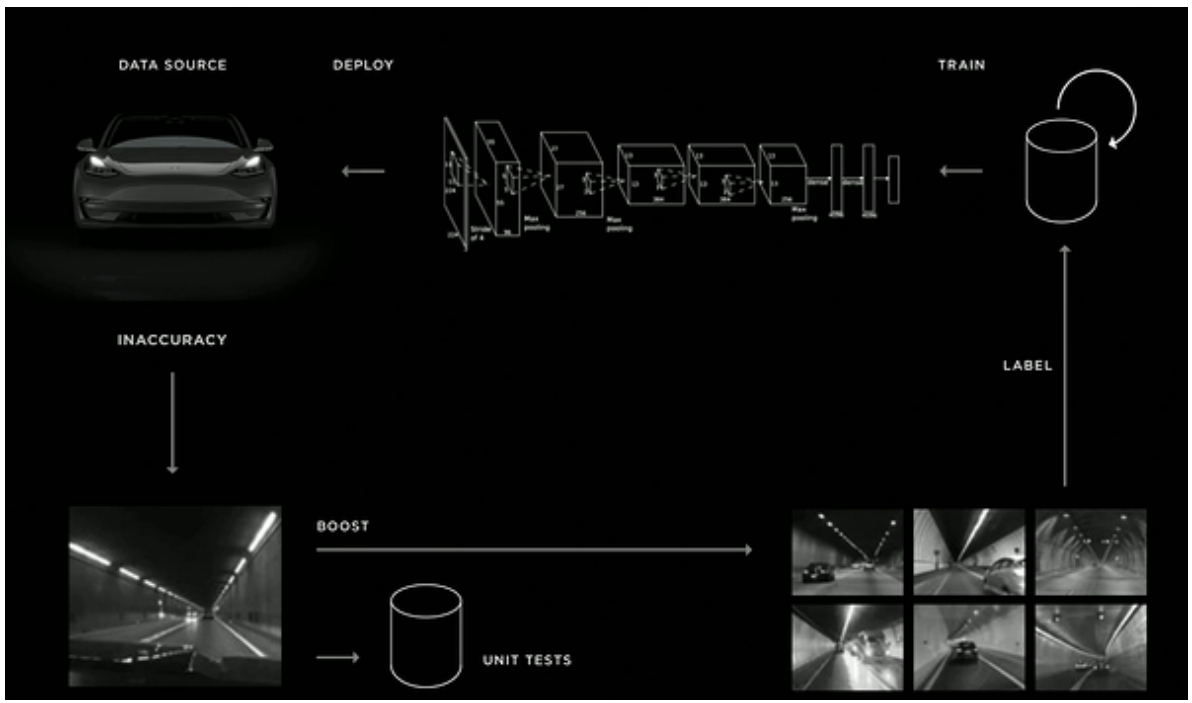
Lifecycle of a machine learning model

Data can induce smaller sub-cycles in the model's lifecycle.



The data flywheel

Entire strategies can be defined around the availability of data. In a **data flywheel**, models are never finished: deployment produces new data that is used towards improvement, followed by a re-deployment.



Tesla Data Engine: `https://www.youtube.com/watch?v=Ucp0TTmvqOE`

Agenda

Machine learning models as products

Deployment options

Machine learning system design

ML project management

Machine learning modelling in context

The role of the deployment stage depends on the specific context of the machine learning project. Deployment can be a **goal**, but also seen as one component of the development strategy.

Common scenarios include:

- **Fundamental research**: Concerned with profound understanding, wide and open impact, very long timescales, 1 cycle / model.
- **Analytics**: Insight, exploration, reporting and decision making, impact on organisations, different timescales, 1 cycle / model.
- **Commodified component**: Commercial orientation, outsourced, long timescales, stable, 1 cycle / model.
- **Strategic component**: Commercial, viability, short (e.g. Netflix) to long (e.g. Tesla) timescales, dynamic, many cycles / model.

Pipelines, not models

From a data processing perspective, machine learning systems cover the entire data pipeline, from raw data to final action, not just the machine learning model:

- Data ingestion
- Data preparation.
- Feature extraction.
- ML model.
- Action.

In general, we will deploy machine learning **pipelines**, not just models. Do not confuse pipelines with workflow!

A **reproducible** deployment is one where the built and deployed machine learning pipelines are such that they produce the same output when they are presented the same input.

The notion of environment

A computing environment consists of a collection of hardware and software tools that are integrated to work well together.

Data science environments are specifically designed to process and analyse data. This specialisation can be achieved by including specific libraries (e.g. NumPy, pandas, Scikit-learn, etc).

A **deployment** (or **production**) **environment** is the target environment where machine learning models are used. Data science environments can be used for deployment but in general, we could have **any** computing environment, which can make deployment very difficult.

Deploying in data science environments

Models can be deployed in the data science environment where they have been built. In this case, the data science environment provides both the building and the deployment runtime. Typical scenarios include:

- Research.
- Analytics.

After training a model, it can be *persisted* by using **serialisation**, which converts the model into a stream of bytes that can be stored and from which the model can later be extracted.

Note: Examples of serialisation options in Python include pickle, joblib's dump, Tensorflow's save or Matlab's save. Open standards for model serialisation that are language-agnostic include PMML, PFA, ONNX.

Deploying in other environments

The deployment environment will in general be different from the data science environment where models were originally built.

Even when the deployment environment is of the same nature, running a machine learning model and a pipeline requires additional software components (i.e. **dependencies**) that could have different versions.

Executables

Data science environments are **high-level interpreted** systems and many target devices will be able to run **executable files** only.

Machine learning models can be embedded in an executable file (the final application):

- Directly creating an executable file (e.g. using Pyinstaller).
- Generating low-level code (e.g. C). Then embed in another low-level program and compile into an executable.

Embedding a machine learning model in an executable file brings on maintainability challenges. An option is for the application to call an external service that runs the model and serves the predictions.

Packaging

Packaging can be used for deployment purposes but also with a focus on reproducibility and collaboration. Packaging options include:

- Models as dependencies (e.g. Python modules).
- Models and other dependencies in the environment (e.g. conda).
- Containers (e.g. Docker) packaging code, runtime, dependencies as as *images* (not pictures!).

Packaging the environment (conda and docker) ensures that the model and its environment are replicated exactly when deployed.

Cloud deployments

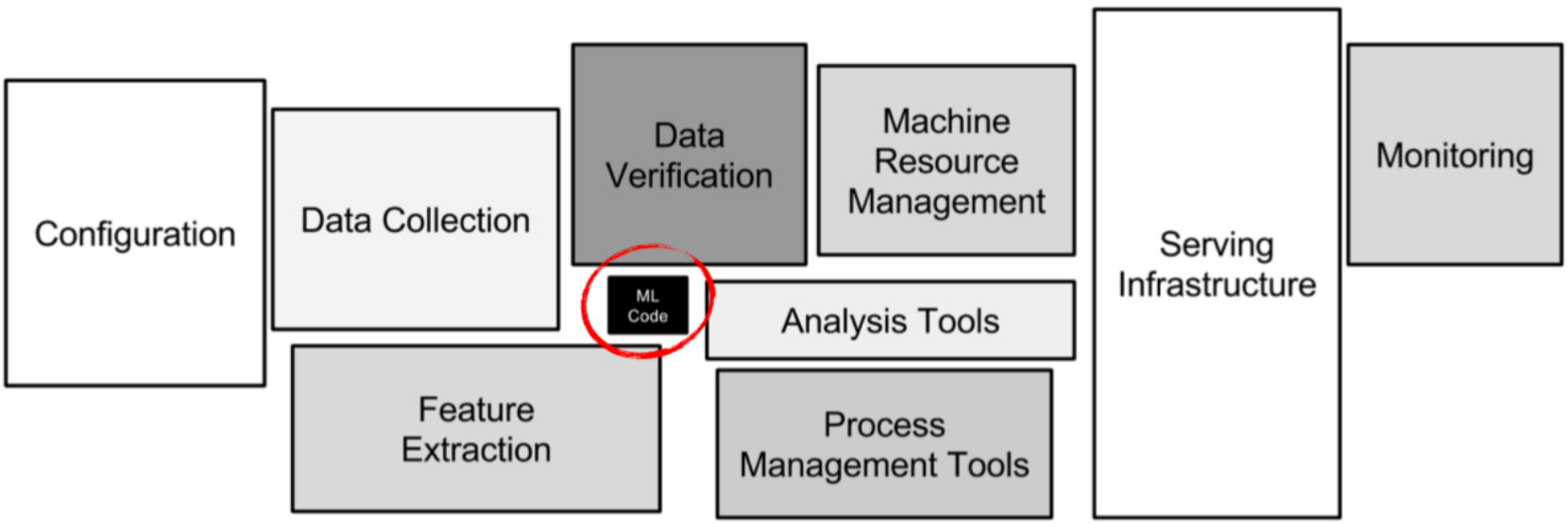
Machine learning models can be deployed on the cloud, i.e. on servers that are accessible via the internet. This option is the default one when **scalability** is a must.

Models can be hosted on the cloud as serialised objects or containers and predictions can be stored or returned using APIs. Cloud platforms offer services with different levels of control, flexibility and management:

- Compute services.
- Container services.
- App hosting (e.g. AWS Elastic Beanstalk, Google App Engine).
- Serverless (e.g. AWS Lambda, Google functions).
- ML specific services (e.g. AWS SageMaker, Google AI).

Machine learning systems: More than machine learning

In many scenarios, the cost of non-ML components in machine learning systems can be much higher than the cost of the actual ML component.



https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

Agenda

Machine learning models as products

Deployment options

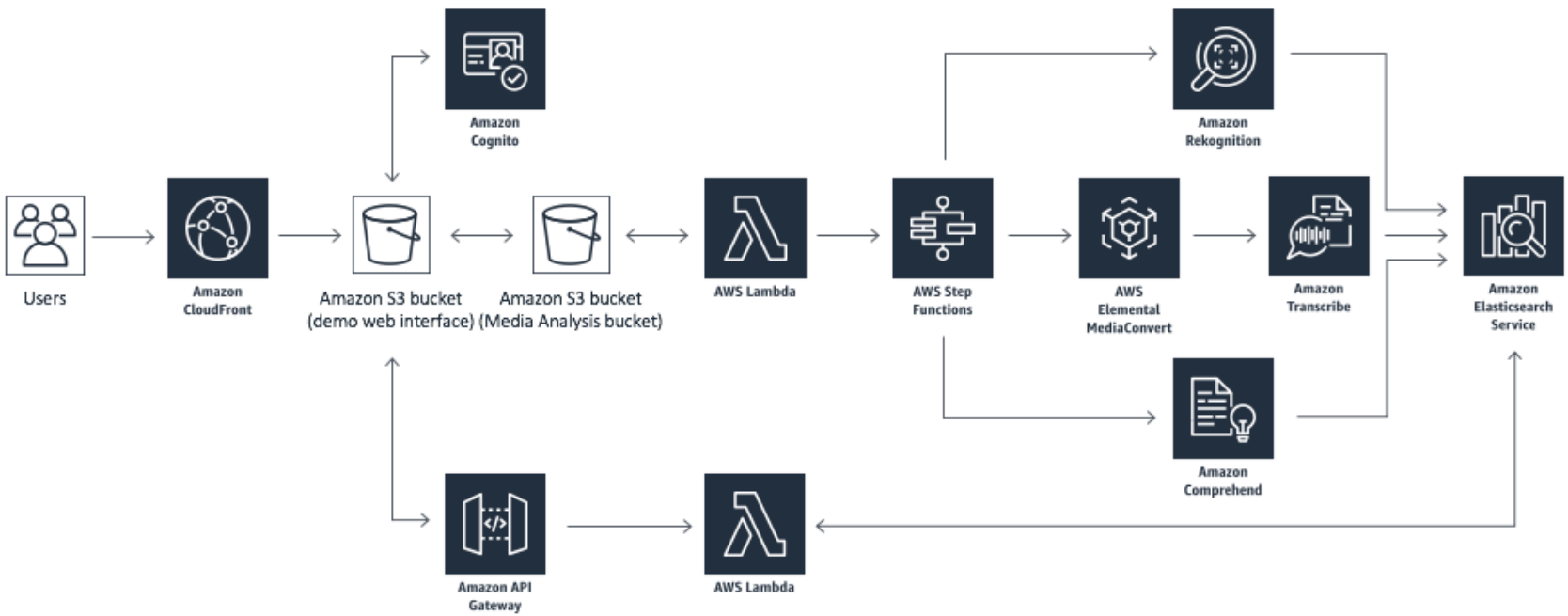
Machine learning system design

ML project management

Machine learning systems

Machine learning systems are more than a machine learning model, they consists of infrastructure (hardware, networking, OS), application software, data and other components.

The architecture of a machine learning system describes how its software components are arranged and interact with each other.



Machine learning system specs

Machine learning system design includes **non-statitical** requirements that are irrelevant from a machine learning perspective:

- Highest accuracy (data science team).
- Integrability, reliability, scalability, maintainability (data engineering team).
- Fast predictions (sales team).
- Maximise profit (managers).

System level requirements can be incorporated as non-statistical constraints during model design.

Accuracy vs speed

Fast, inaccurate predictions are sometimes preferred to slower, more accurate ones:

- Google (2009): Increasing latency from 100mx to 400 ms reduces searches by 0.2 % to 0.6 %.
- Booking.com (2019): 20 % increase in latency costs 0.5 % conversion rate.
- Cloud-based Google Translate vs device-based World Lens: users rated the World Lens results as more accurate and satisfying.

Accuracy vs speed

Faster models can be designed by:

- Choosing faster families of models.
- Making models faster (inference optimisation).
- Making models smaller (compression, e.g. quantisation, pruning, factorisation).
- Using more powerful hardware.

In general, prediction accuracy and speed are conflicting goals.

Memory footprint

Target devices have different memory sizes, as small as kilobytes. Model size considerations might need to be explicitly formulated.

The final size of a model depends on factor such as:

- Number of attributes.
- Model family and architecture (e.g. tree, kNN, SVM, neural net).
- Availability of reduction techniques (e.g. tree pruning).

TinyML is a machine learning area that focuses on deploying machine learning models on constrained systems (**embedded systems**).

Test in production

Pre-deployment testing is needed to understand how a system will behave when deployed, but might not be representative.

Several strategies exist to evaluate machine learning in **real conditions** during deployment:

- **Canary testing** releases a new model to a limited customer group. It reduces deployment risks and allows for gradual replacement.
- **Shadow testing** deploys a model in parallel, but predictions are logged, not served, to be analysed.
- In **A/B testing** is used to compare two or more versions by deploying them simultaneously.

Agenda

Machine learning models as products

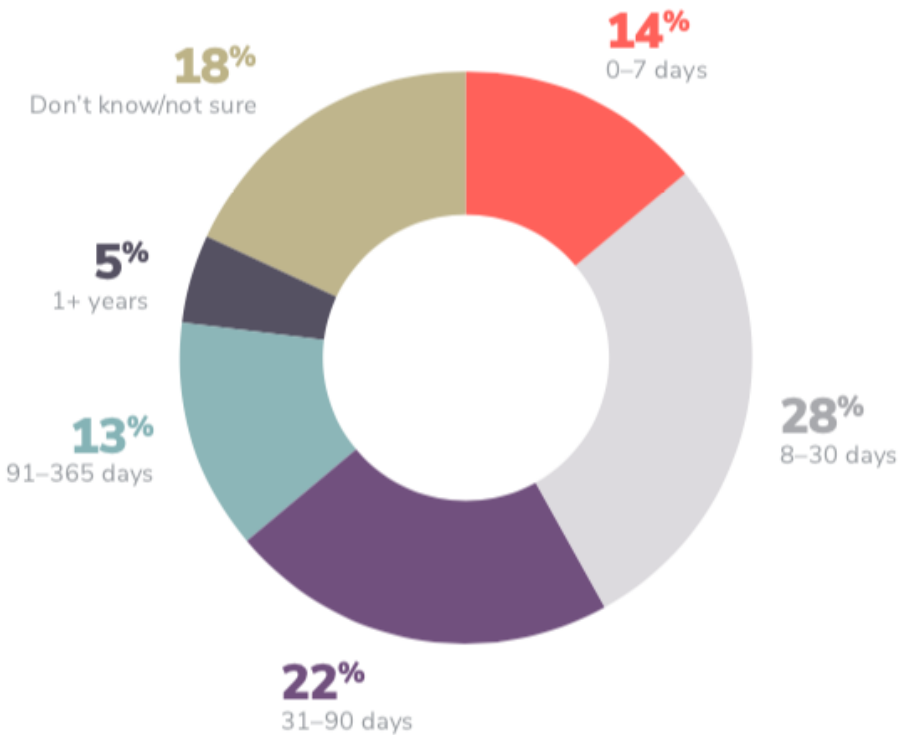
Deployment options

Machine learning system design

ML project management

Machine learning deployment time: A survey

Many tech companies deploy thousands of new models every day. Others report that deploying one single model takes them:



Source: 2020 state of enterprise machine learning (Algorithmia, 2020)

Managing machine learning projects

Machine learning projects, big and small, involve a large number of iterations that result in multiple prototypes. Managing these cycles is in general a good practice and in many cases unavoidable.

Software engineering management can be readily applied to machine learning projects. Data plays however a special role, which is absent from the typical software project:

- What data has been used to train this model?
- Which models have been trained using this data?
- Which deployments need to be updated if we refresh this data?

Versioning and tracking

To manage machine learning products, versioning is essential as it allows to record and track models and associated information produced during each iteration.

Specifically, it is important to apply versioning to:

- Data.
- Code (model building, defining pipelines).
- Models.
- Environment.

There exist tools to automatically manage machine learning projects and facilitate collaboration (e.g. Data Version Control). Experiments can be organised, variations can be tracked and results can be compared.

MLOps: End-to-end machine learning management

A number of platforms have been recently developed for managing the end-to-end machine learning lifecycle, from data preparation, through model building to deployment and monitoring.

Key features of these platforms include:

- Model versioning and collaborative development.
- Experiment tracking and comparison.
- Code packaging in a reproducible way, for sharing and deploying.
- Model deployment, from ML libraries to model serving and inference platforms.

Databrick's **mlflow** is one example of such platforms.