Machine Learning and Data in Operation

Bertil Chapuis

Beat Wolf

Who are we?

- Chapuis Bertil
  - bertil.chapuis@hes-so.ch
  - PhD in distributed systems
  - Teaching software engineering and project management

- Beat Wolf
  - beat.wolf@hefr.ch
  - PhD in computer-science, application domain of bio-informatics, genetics
  - Teaching ML, Data Analysis and programming

Moodle & Teams

- Moodle:
  - https://moodle.msengineering.ch/course/view.php?id=**2793**
  - Key: **MLOps24**
- Teams:
  - Link on Moodle or code: **qavum83**

Objectives

- Recognising the **complete lifecycle of machine learning projects**, from data requirements to development, deployment, and monitoring.
- Demonstrating skills in **maintaining ML code and data, version and integrate it, and define appropriate environments**, with emphasis on practical applications such as data cleaning and preprocessing.
- **Deploying ML models at scale, monitoring their performance and adapting models to changing requirements**, with a focus on assessing and adjusting to data drift and shifts in data distribution.
- **Analysing relevant tools and real use-case scenarios, such as real-time services management**; critically analysing the implications and applications in practical scenarios.
- **Selecting software and hardware platforms** based on the requirements of different scenarios, demonstrating a thorough understanding of the needs and constraints of each.
- Extracting and integrating insights from guest lectures by industry professionals (subject to availability), **demonstrating the ability to interpret expert knowledge from scientific literature and online resources**, and applying it effectively to complement their hands-on experience.

Course overview

- Brief recap of machine learning and deep learning
- Introduction to the lifecycle of a Machine Learning project
- Understanding data needs and requirements for ML projects
  - Versioning, storage, processing, labeling, augmentation, simulation
- **ML Development**
  - Defining the environment
  - Maintaining the ML code
  - Integrating ML code (versioning, evaluation, baselines).
- **ML Deployment**
  - Running models at scale (e.g. batch vs online, model compression, cloud / edge deployment)
  - Ensuring system availability
  - Monitoring performance
  - Adapting to changes (data distribution shifts, failures, metrics, logging, continual learning).
- Exploration of tools and real-world scenarios
- Overview of relevant hardware and software platforms
- Advanced topics such as guest lectures, project management etc.

Semester plan

- 2 Introduction
- 3 MLOps
- 4-5 MLOps guide
- 6 Advanced MLOps topics
- 7 Kickoff project
- 8-9 Group project
- **10 Seminar presentation**
- 11-13 Group project (possible guest lecture)
- **14 Final presentation**

Marks

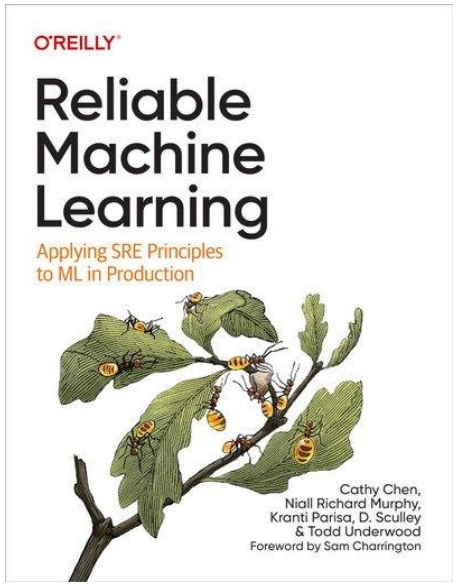
- Group project (maximum 4 people) 60%
  - 20% Seminar presentation
  - 20% Final presentation
  - 20% Project (Code, documentation etc.)
    - The students are responsible to assure the contribution is balanced
- Written exam (40%)
  - Covering both the theory seen in class and the seminars

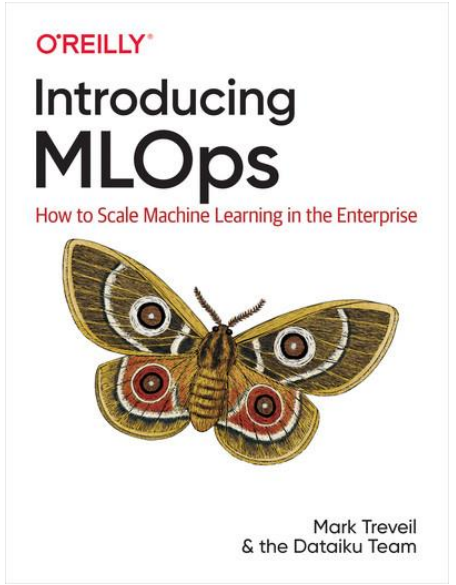
Group project

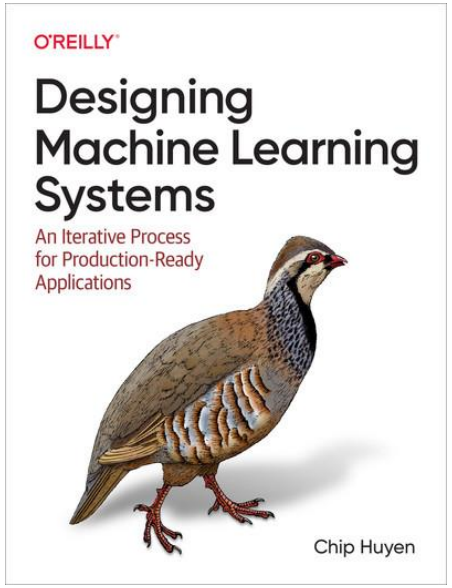
- Every group will explore a MLOps and/or DataOps specific problem
- A proof-of-concept project around the thematic will be developed and deployed
- A first presentation explores the theoretical aspects
  - Included in the final evaluation
- The second presentation focuses on the practical aspects of the project

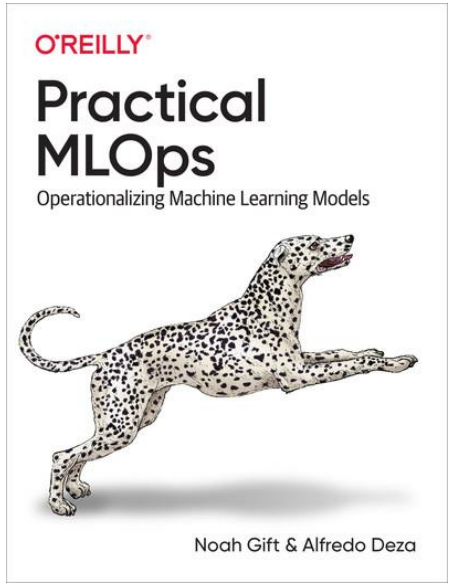
Resources

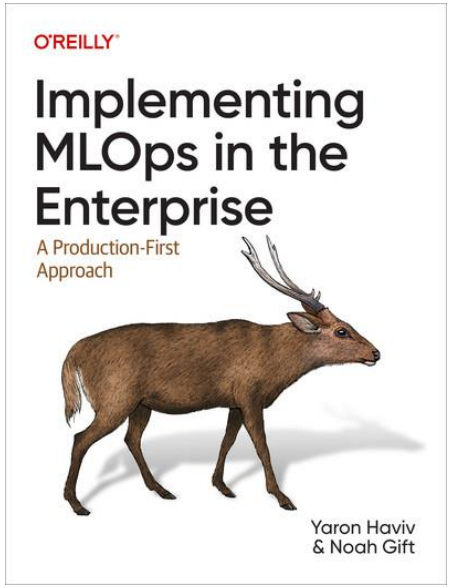
- A small selection of relevant resources











Discussion on experience

Project discussion

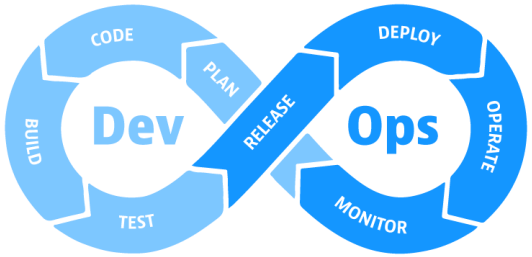
- Form 4 groups and discuss:
  - What data analysis and machine learning projects did take part of until now?
  - What were your main problems in those projects?
  - Would you now approach the projects differently?

- One person of the group will present a condensed version of the discussion to the class

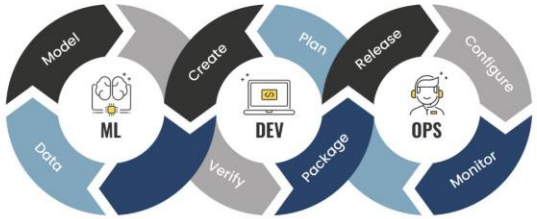
What is data analysis?

DevOps, DataOps and MLOps

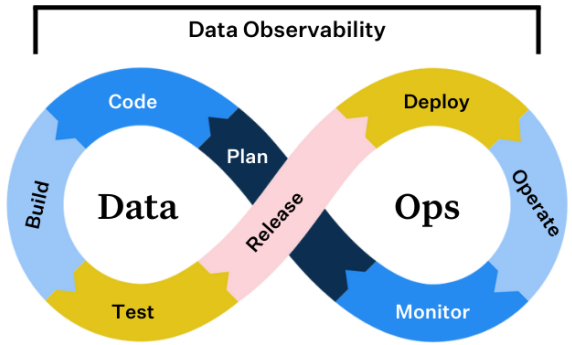
- DevOps has been the defacto standard of developing and deploying software for some years
- We look at the extensions of that concept **DataOps** and in particular **MLOps**
- How do we go from a simple data analysis project to a deployed state of the art machine learning solution?



https://www.dynatrace.com/news/blog/what-is-devops/



https://ubuntu.com/blog/what-is-mlops



https://www.montecarlodata.com/blog-what-is-dataops/

Data analysis

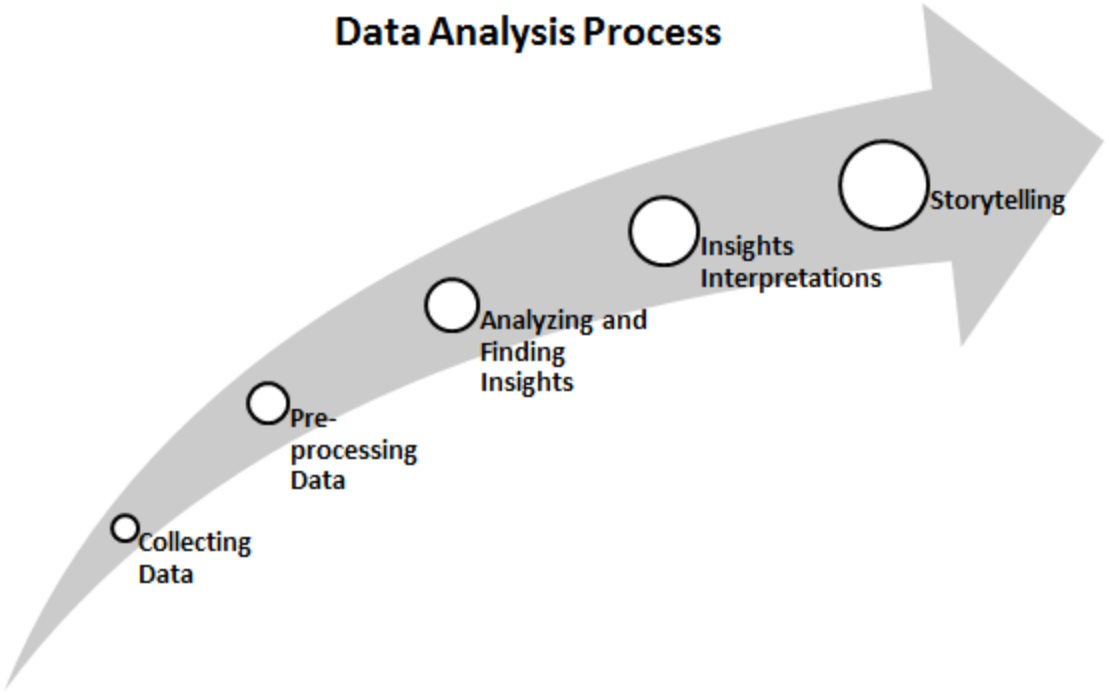
- At the core of all MLOps and DataOps projects, there is a Data Analysis project

- **Wikipedia** (https://en.wikipedia.org/wiki/Data_analysis) :
- **Data analysis is the process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making**.[1] Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains.[2] In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively.[3]

- We want to extract relevant information from raw data to:
  - Predict the future
  - Understand the past
  - Find anomalies
  - Develop new features
  - Etc.

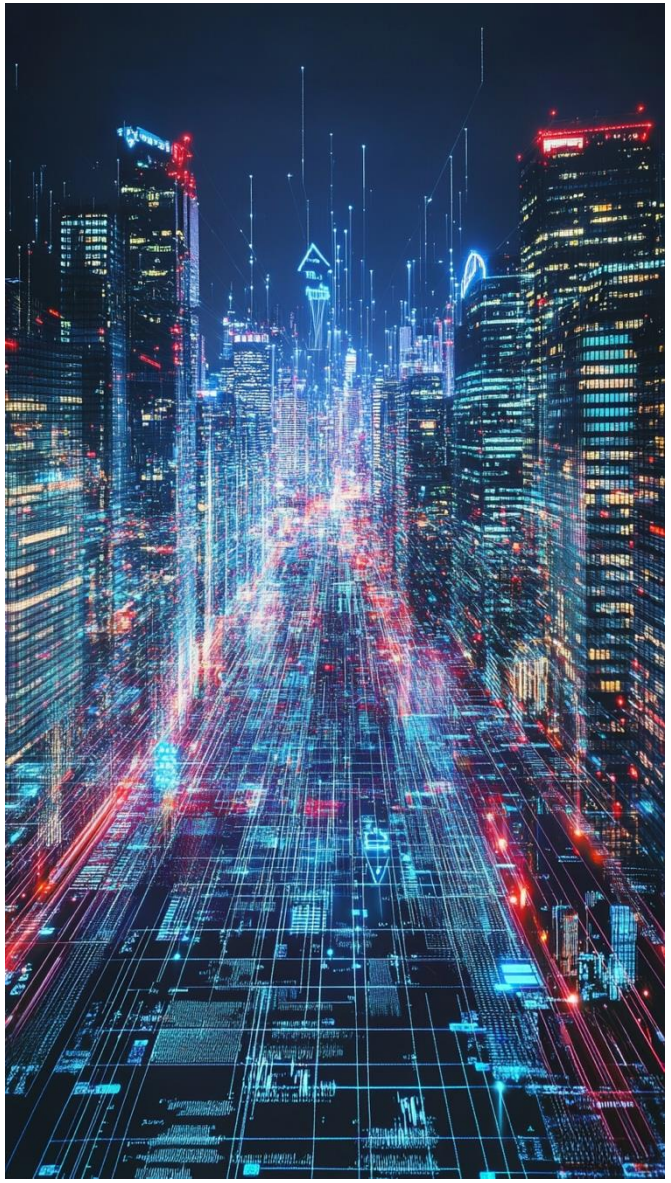
Data analysis steps

- Data analysis projects have standard steps
  - Data collection
  - Data pre-processing
  - Analysis
  - Interpretation
  - Storytelling
    - Reports, dashboards etc.



Big data

- The principal challenges of Big Data are
  - How to retrieve data
  - How to clean the data
  - How and where to store it
  - How to search through the data
  - How to transfer data (or parts of it)
  - How to perform data analysis
  - How to visualize it

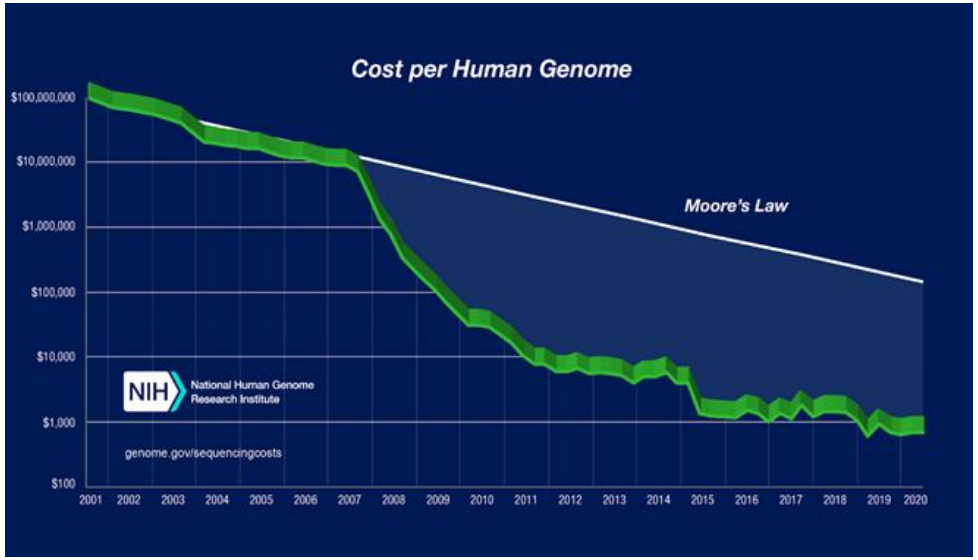


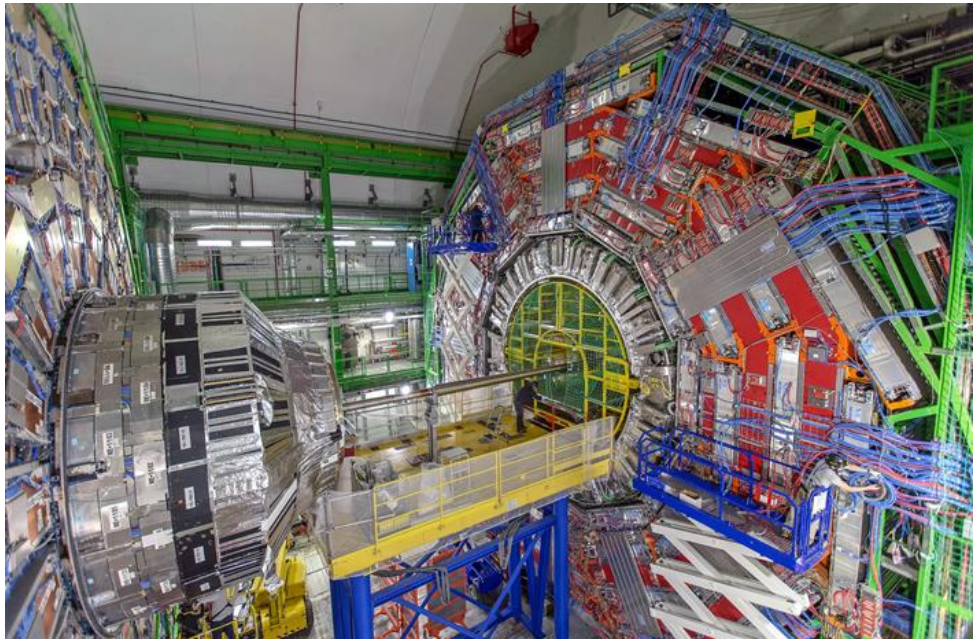
The 4 (+1) V of Big Data

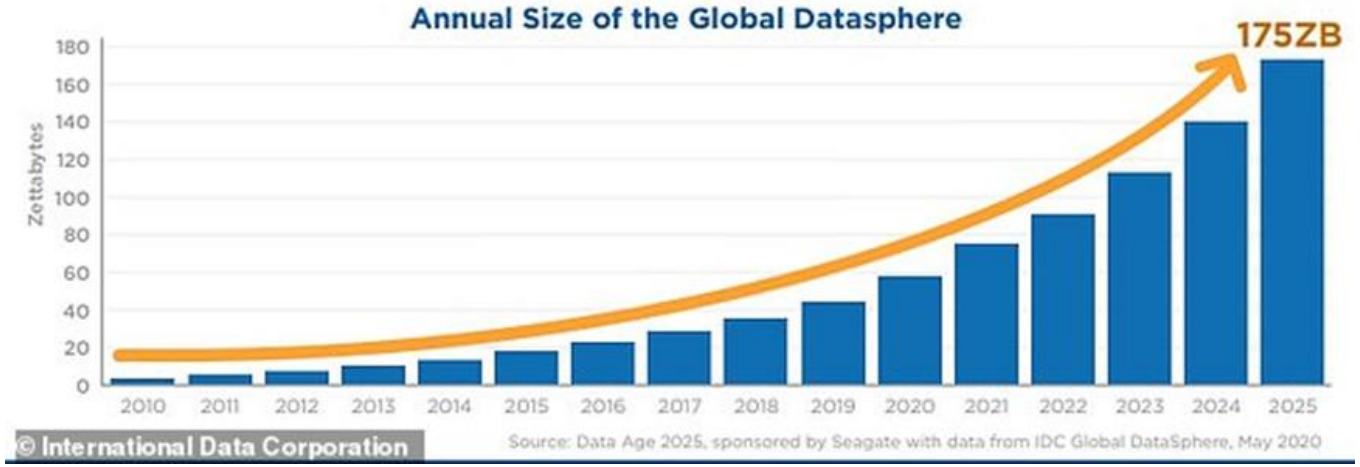
- There exists a more precise definition of Big Data, which uses 4 attributes characterize it:
  - **Volume**
    - The amount of data
  - **Variety**
    - The type and heterogenity of the data
  - **Velocity**
    - The frequency of change
  - **Veracity**
    - The data quality
- The Bonus V, is the result of Big data
  - **Value**
    - The value generated through big Data Analysis

Volume

- The volume of data generated is steadily increasing
- Most of the data humanity produced has been created in the last few years
- Creating data ins increasingly cheap
- Classic tools are no match for this type of data

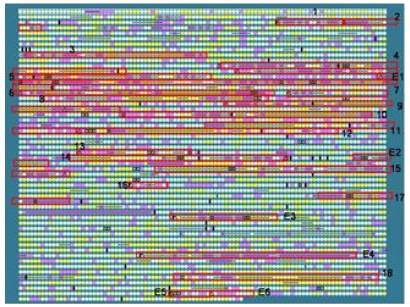


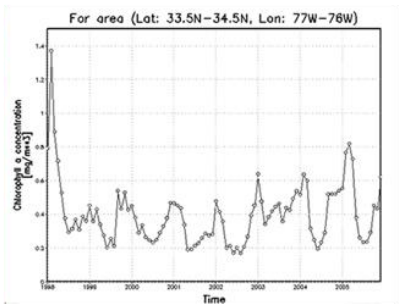


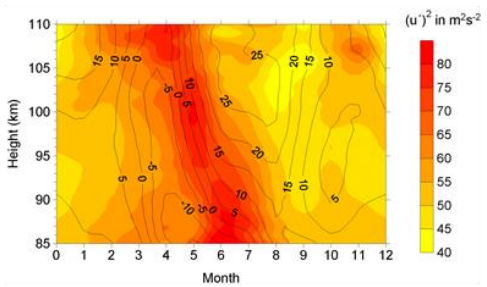


Variety

- Data can come in a large variety of formats and from very different sources
- Often the combination of the various sources gives true value
- Many classic tools are good at one or two datatypes, but programming is required to handle them all
  - Text, numerical, images, videos, sound, tabular data, graph data etc.







Velocity

- Data is produced in real time
  - Not at the end of the day, week or month
- Real time sensory data, sales, stock market etc. also need immediate reactions
- Classic tools are usually focused on offline data analysis



Veracity

- This fourth V is a bit of a bonus, but crucial
- Considering the 3 other Vs (Volume, Variety, Velocity) it is very hard to guarantee the quality of the data
- The data can be wrong, or incoherent between sources for many reasons
- Handling those quality issues at scale is key to (big) data analysis

Value

- The 4 Vs together: Volume, Variety, Velocity and Veracity lead to key insights, the Value extracted through Big Data analysis
- The hope is that this leads to better sales, higher RoI, less production issues etc.
- Handling data at this volume, heterogeneity and speed requires appropriate tools
  - Automation and scaling in particular
- Data analysis and machine learning are key tools to achieve those goals

Methods

- Data analysis projects have many steps
- One key step is the **E**xploratory **D**ata **A**nalysis (EDA)
- The role of the EDA is to have a better understanding of the data

    - The exact content
    - Quality (missing data, duplicates, outliers)
    - Suitability for project goals
    - Potential limits of the data

EDA overview

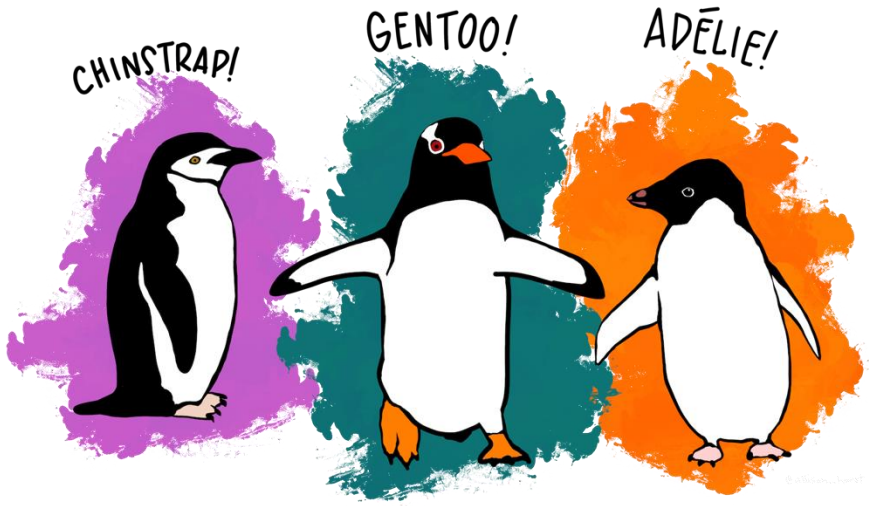
- The EDA is an iterative and creative process
- A series of questions is asked and answered
  - With some answers generating new questions
- Tools like plots, statistical analysis etc. are used to ask increasingly complex questions
  - What data columns and how much data do I have?
  - Are there missing or duplicate values?
  - Are there any correlations between my columns?
  - Can past values be used to predict future values?
  - Etc.

EDA results

- At the end of an EDA there is a report that
  - Precisely describes that data and its content
  - Clearly indicates the quality shortcomings
    - And thus, future quality assurance strategies
  - Describes how adapted the data is for our purposes
  - Gives first models (statistics or machine learning) to solve the problem
- The EDA is a Go/No go step in a data analysis project
  - And especially interesting because it contains all data analysis problems in a condensed form

Example EDA

- We look at a small dataset about pinguins
- It is small, but has some interesting properties
- The underlying goal is to predict the species of the penguin
- We will quickly go through the different steps of an EDA



Descriptive statistics

- We describe the dataset on a high level
  - How does the content look like?
  - How much data?
  - Column types
  - Overview statistics

```
len(df)
```

```
344
```

```
df.dtypes
```

	0
rowid	int64
species	object
island	object
bill_length_mm	float64
bill_depth_mm	float64
flipper_length_mm	float64
body_mass_g	float64
sex	object
year	int64

```
df.head()
```

	rowid	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	1	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	2	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	3	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	4	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
4	5	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

```
df.describe()
```

	rowid	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	344.000000	342.000000	342.000000	342.000000	342.000000	344.000000
mean	172.500000	43.921930	17.151170	200.915205	4201.754386	2008.029070
std	99.448479	5.459584	1.974793	14.061714	801.954536	0.818356
min	1.000000	32.100000	13.100000	172.000000	2700.000000	2007.000000
25%	86.750000	39.225000	15.600000	190.000000	3550.000000	2007.000000
50%	172.500000	44.450000	17.300000	197.000000	4050.000000	2008.000000
75%	258.250000	48.500000	18.700000	213.000000	4750.000000	2009.000000
max	344.000000	59.600000	21.500000	231.000000	6300.000000	2009.000000

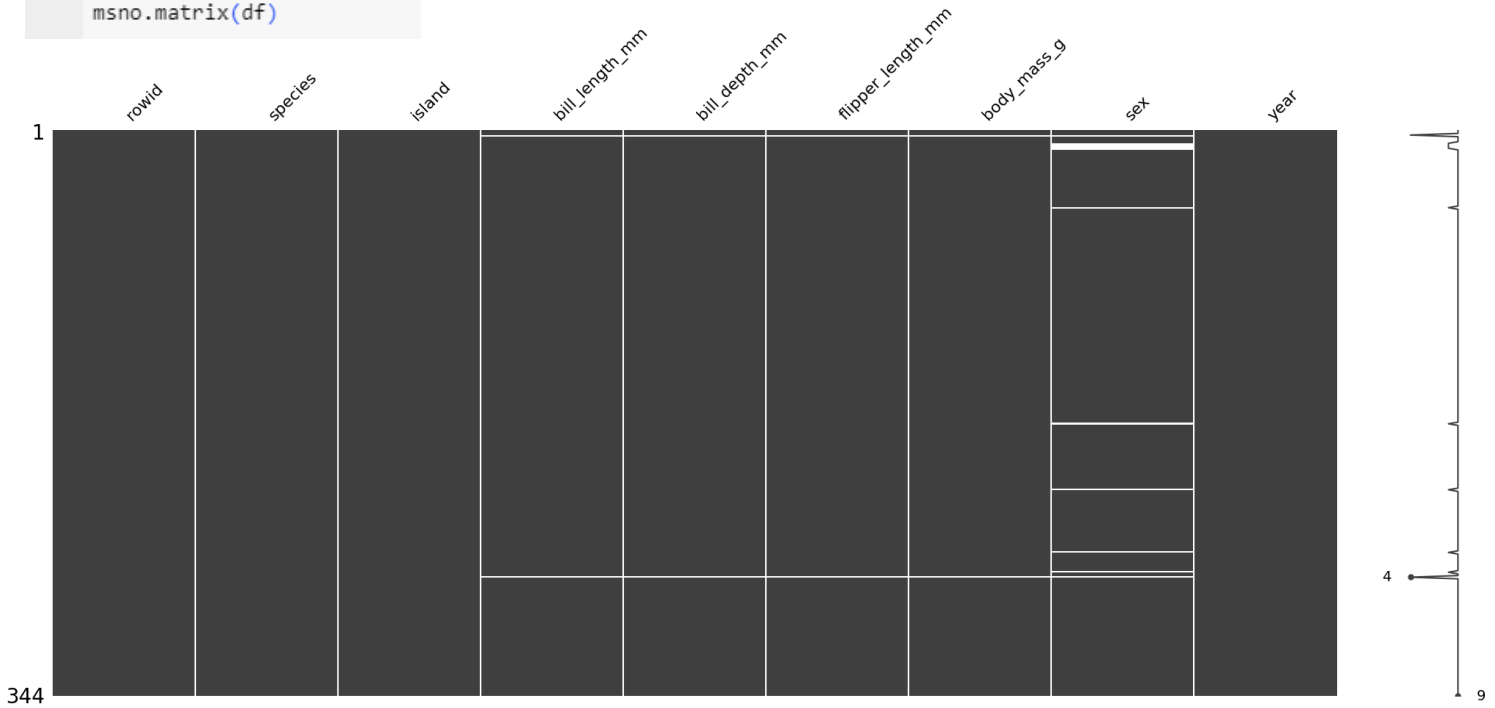
```
df.describe(include='object')
```

	species	island	sex
count	344	344	333
unique	3	3	2
top	Adelie	Biscoe	male
freq	152	168	168

Data quality

- We check for missing data and outliers
  - No particular outliners have been found

```
import missingno as msno
msno.matrix(df)
```

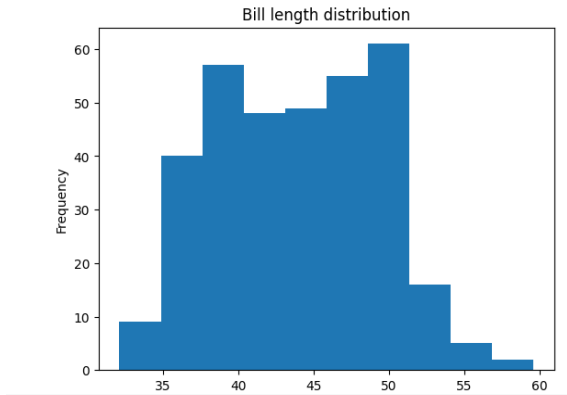


Single columns

- We look at the distribution of the different data columns
- What type of distribution do they have?
  - Over/underrepresented values
  - Outliers
  - Etc.

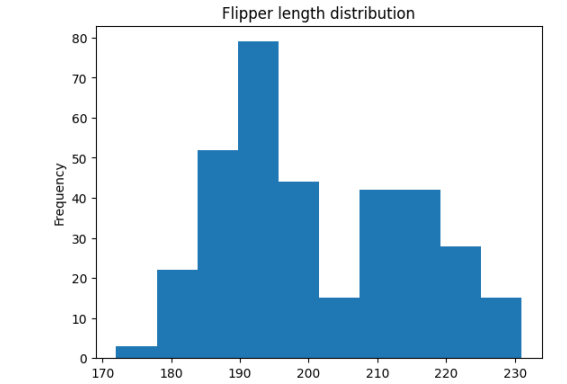
```
df["bill_length_mm"].plot.hist(title = "Bill length distribution")
```

```
<Axes: title={'center': 'Bill length distribution'}, ylabel='Frequency'>
```



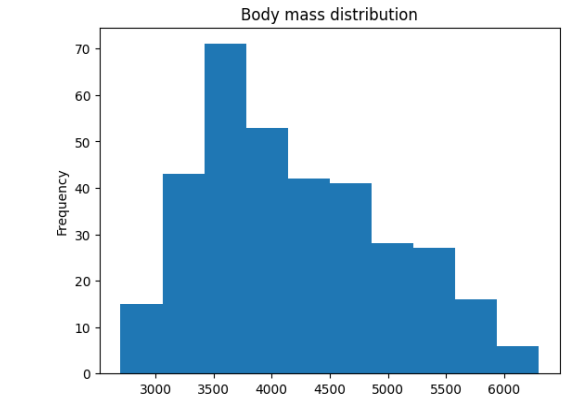

```
[46] df["flipper_length_mm"].plot.hist(title = "Flipper length distribution")
```

```
<Axes: title={'center': 'Flipper length distribution'}, ylabel='Frequency'>
```



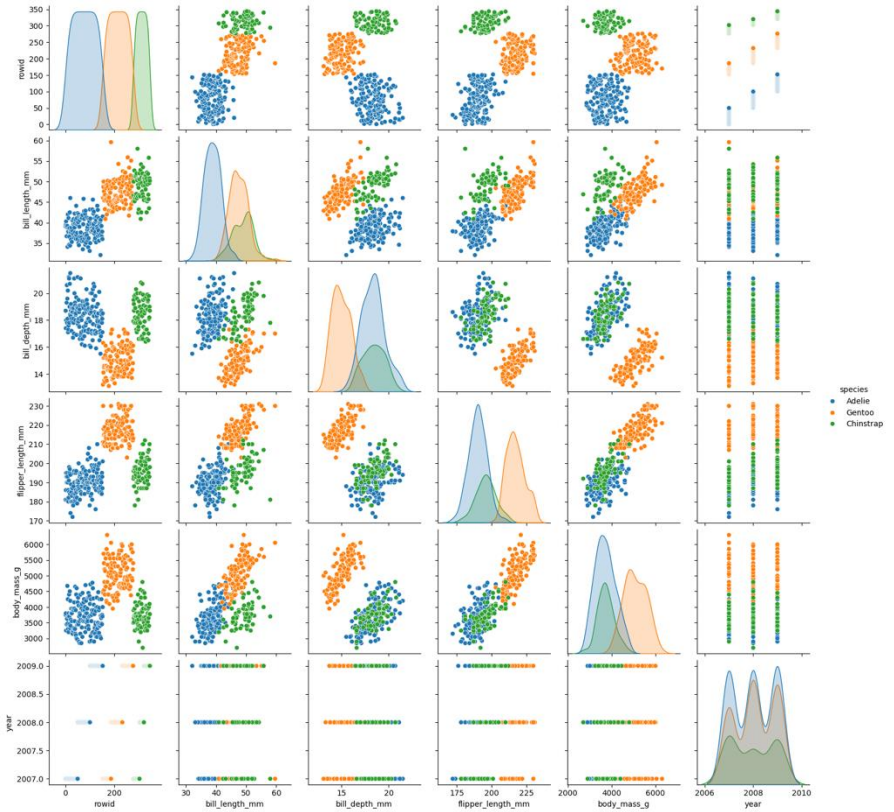

```
df["body_mass_g"].plot.hist(title = "Body mass distribution")
```

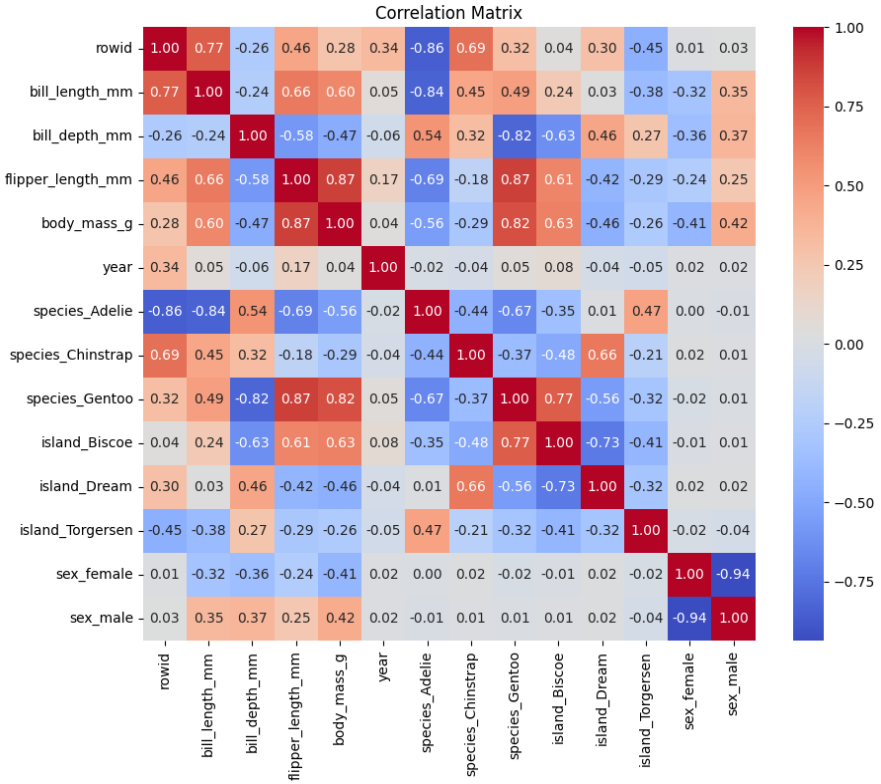
```
<Axes: title={'center': 'Body mass distribution'}, ylabel='Frequency'>
```



Multi column

- More interesting is the multi column analysis
- We also check for simple correlations between columns





Simple baseline

- The EDA usually ends with a simple model to evaluate the potential of the data
- At this stage we understood the data (transformations needed, quality etc.) and can use this knowledge for the real project


```
# prompt: Simple sklearn models (test a few) to classify the species column in the df_orig datafra

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Drop rows with missing values
df = df_orig.dropna()

# Drop island and rowid and use species as the target
df = df.drop(['island', "rowid"], axis=1)

X = df.drop('species', axis=1)
y = df['species']

# Convert categorical features to numerical using one-hot encoding
X = pd.get_dummies(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", accuracy_rf)
```

```
Random Forest Accuracy: 0.9850746268656716
```

EDA summary

- The exploratory data analysis is a key step of data analysis
- It brings into light:
  - The exact nature of our data
  - Any correlations in the data
  - Quality issues
  - First ideas on how to solve the problem and how likely they are to succeed
- A successful EDA is not yet a maintainable production level application
  - But a good step towards it

Data analysis overview

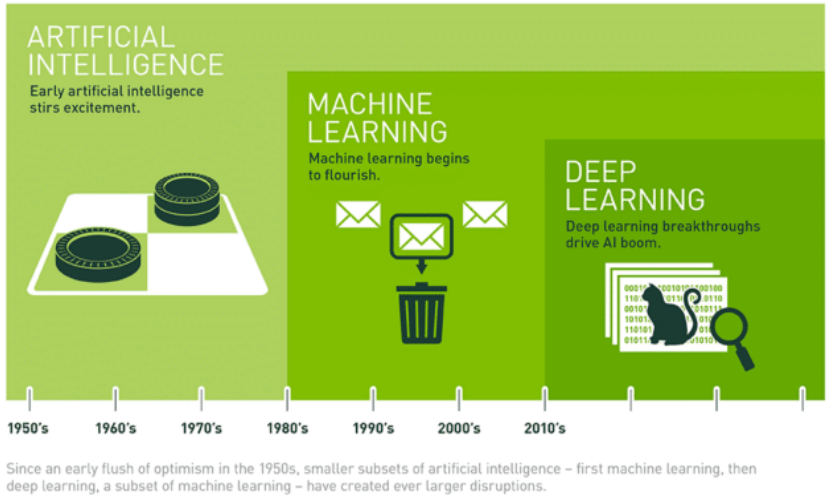
- The goal of data analysis is to go from raw data to actual insights
  - To better understand historical data
  - Better predict future data
  - Find anomalies
  - Etc.
- While doing so we also :
  - Develop a deeper understanding of the data sources
  - Analyse the quality of the data
  - Know the limits of the data

What is machine learning?

Concepts, use-cases and methods

Main machine learning approaches

- Machine learning is a process in which a system learns to perform a task by looking at examples, by using algorithms and statistics, without being explicitly programmed to solve the task at hand.
- A bit more formal
- A computer program is said to learn from **experience E** with respect to some **task T** and some performance **measure P**, if its performance on T, as measured by P, improves with experience E
  - Tom Mitchel – 1998 (https://www.cs.cmu.edu/~tom/)
  - **Measuring performance of systems is key!**

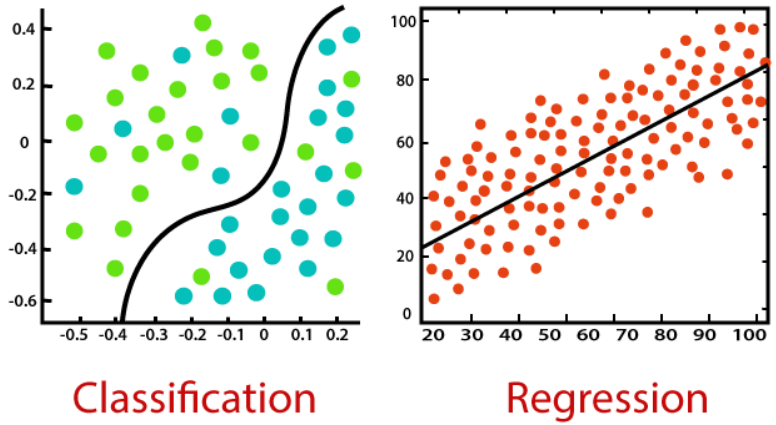


Types of machine learning

- Supervised learning
  - Classification
  - Regression
- Unsupervised learning
  - Clustering
  - Dimensionality reduction
- Generative AI
- Reinforcement learning

Supervised learning

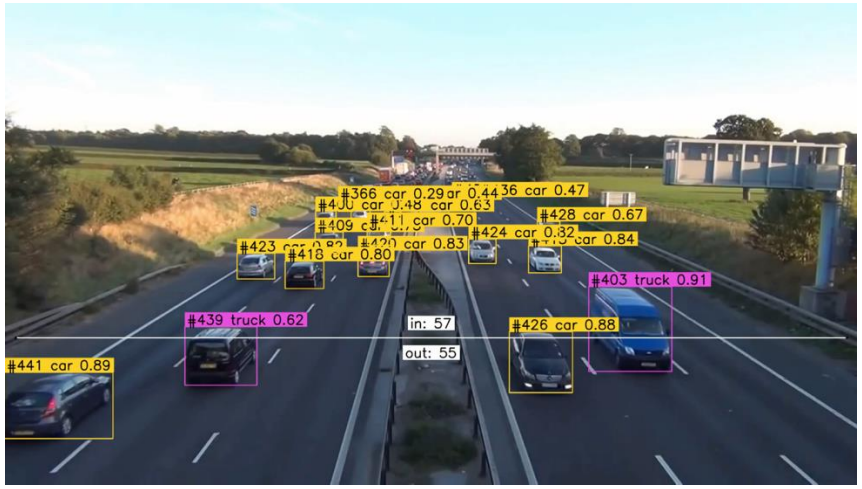
- The classic machine learning approach is supervised learning
- Based on explicit input/output examples, models are trained
- Classification and regression are the classic examples of this
  - Classify documents, images etc.
  - Predict the size or weight of people, stock price etc.

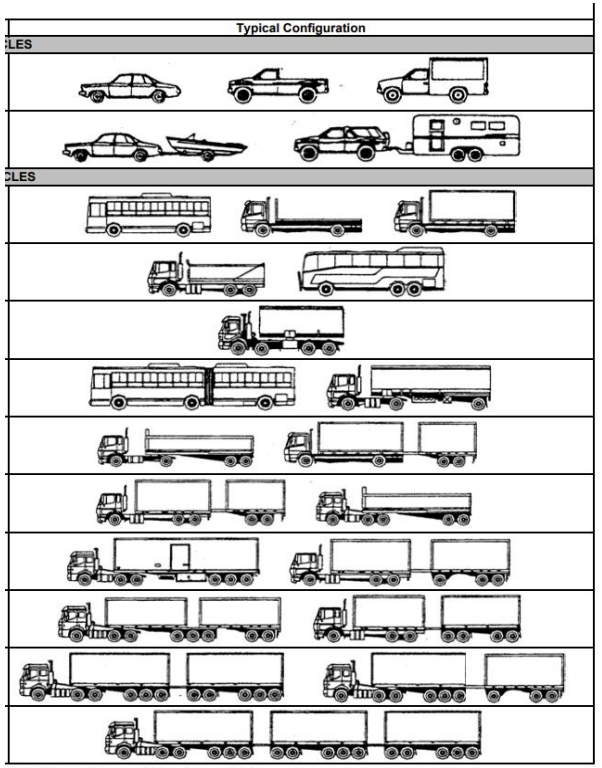


https://www.javatpoint.com/regression-vs-classification-in-machine-learning

Example classification

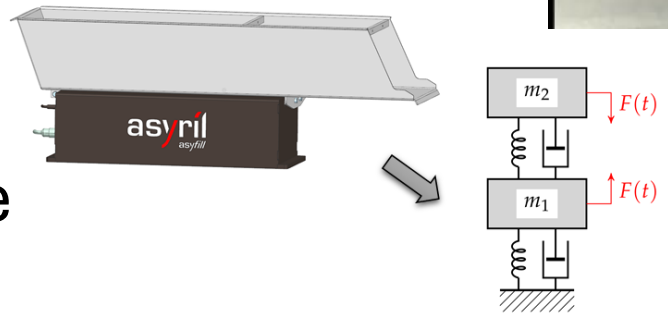
- Knowing what types of trucks are on swiss roads is crucial
- Existing sensors (in the road) are limited in the classification of vehicles
- We developed a method based on Yolo to classify the standard 10 swiss truck categories

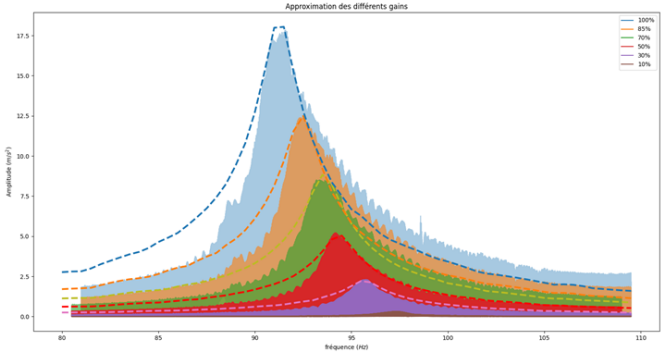




Example regression

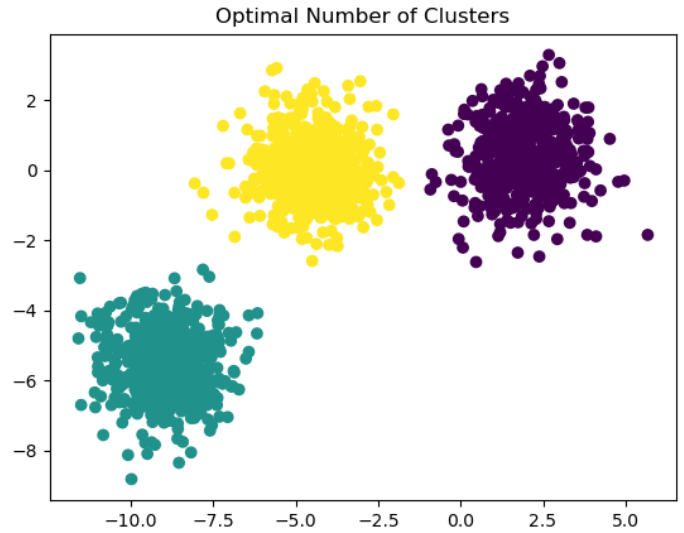
- A vibrating tray is feeding elements to a production chain
- Based on the vibration alone we predict the remaining weight
- Machine learning allows us to use an existing sensor (vibration) to approximate one the machine does not have (weight)



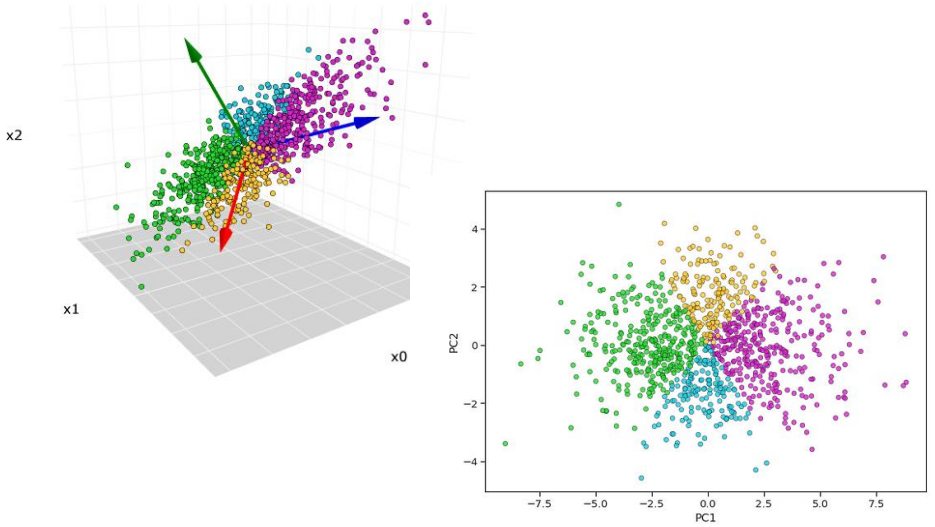


Unsupervised learning

- Unsupervised learning is a very different approach. Without labels we try to identify similar (or unsimilar) datapoints
- The typical examples is clustering
  - We group datapoints into classes, without knowing their true class
- Another one is dimensionality reduction
  - PCA and others reduce dimensions while loosing as little information as possible



https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html



https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d

Example unsupervised learning

- Segmenting handwritten text is a complex problem
  - Identifying the start and end of each character
- While in the end we did not use clustering, the k-means based baseline was much harder to beat than expected
  - Especially after a "smart" initialization of the centroids

rabbit looks at a stoat.

industry will not be

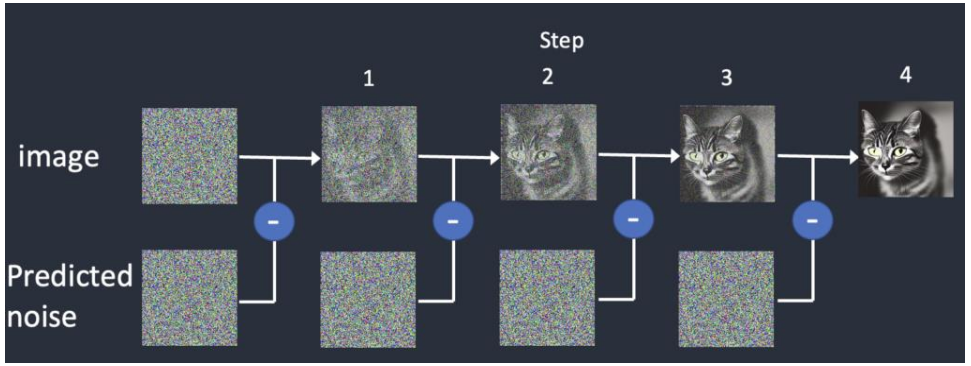
Character Queries: A Transformer-based Approach to On-Line Handwritten Character Segmentation

Michael Jungo[1,2], Beat Wolf[1], Andrii Maksai[3], Claudiu Musat[3], and Andreas Fischer[1,2]

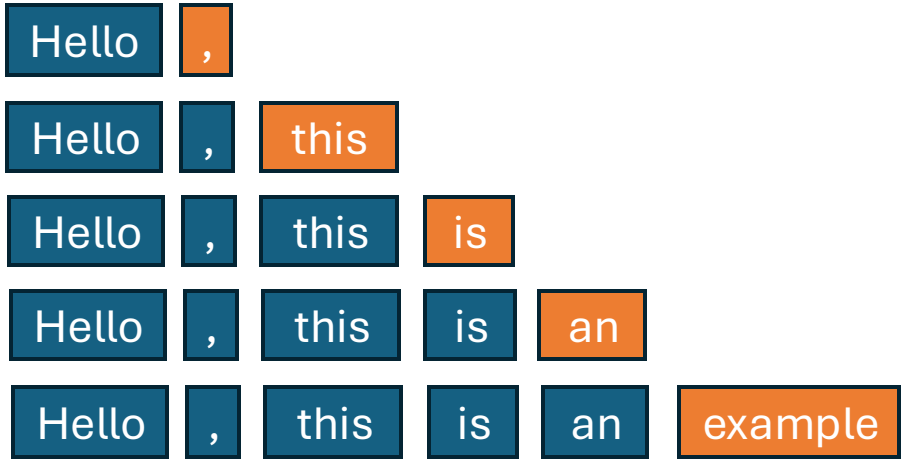
[1] iCoSys Institute, University of Applied Sciences and Arts Western Switzerland
{michael.jungo,beat.wolf,andreas.fischer}@hefr.ch
[2] DIVA Group, University of Fribourg, Switzerland
[3] Google Research
{amaksai,cmusat}@google.com

Generative AI

- Generative AI uses concepts of both supervised and unsupervised learning
  - Next token prediction can be considered unsupervised learning
  - Denoising unlabeled images with no human annotation
- The big particularity of generative AI (LLMs, diffusion models) are that they are iterative
  - They are in general much slower than classic tasks discussed previously

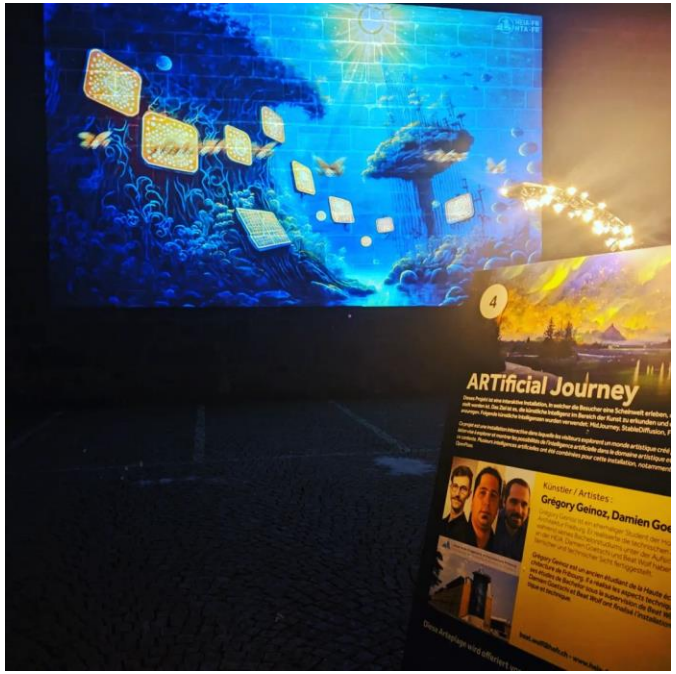


https://stable-diffusion-art.com/how-stable-diffusion-work/



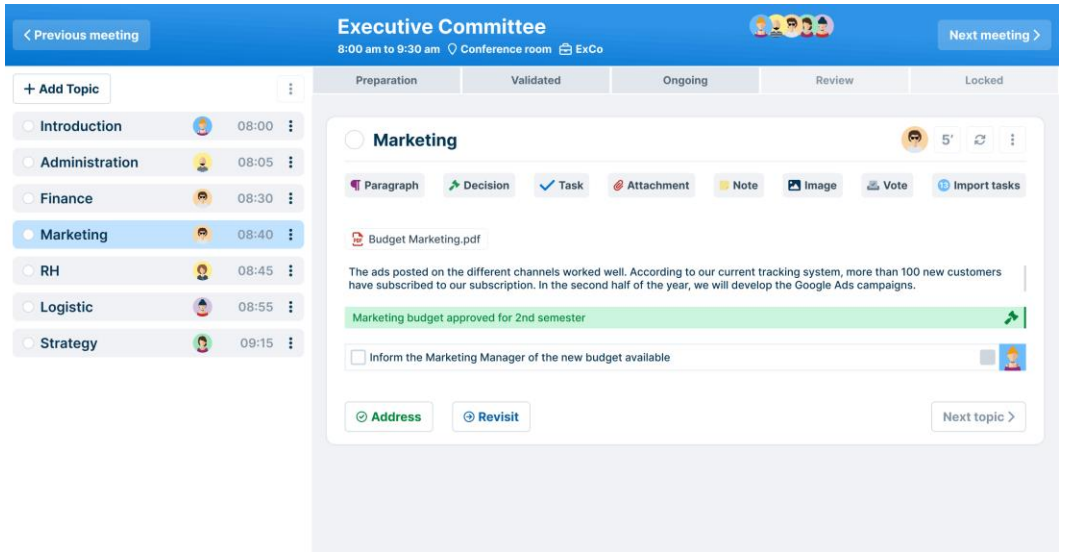
Example generative AI - Image

- We used diffusion-based models to create various art installations
- One example is the interactive art installation at the "Murten Lichtfestival 2022"
- Half an hour of video content has been generated using Midjourney



Example generative AI - LLM

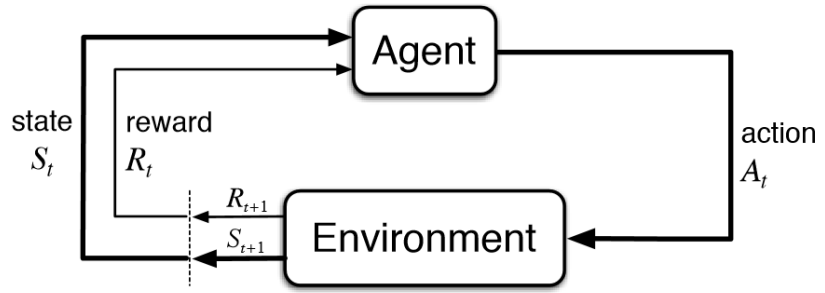
- LLMs are the prime example of generative AI
- We are currently developing a smart assistant for work meetings
  - Agenda preparation
  - Meeting minutes summary
  - Meeting task extraction
  - Etc.
- Privacy aware LLMs and retreival augmented generation (RAG) are at the core of the project



https://wedo.com/

Reinforcement learning

- Reinforcement learning is a different approach yet again
- The system learns to while running by
  - Interacting with an environment
  - Receiving reward signals based on the actions taken
  - Exploring through trial and error different actions
- Models based on this technique are very powerful, but less used in standard ML projects



https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

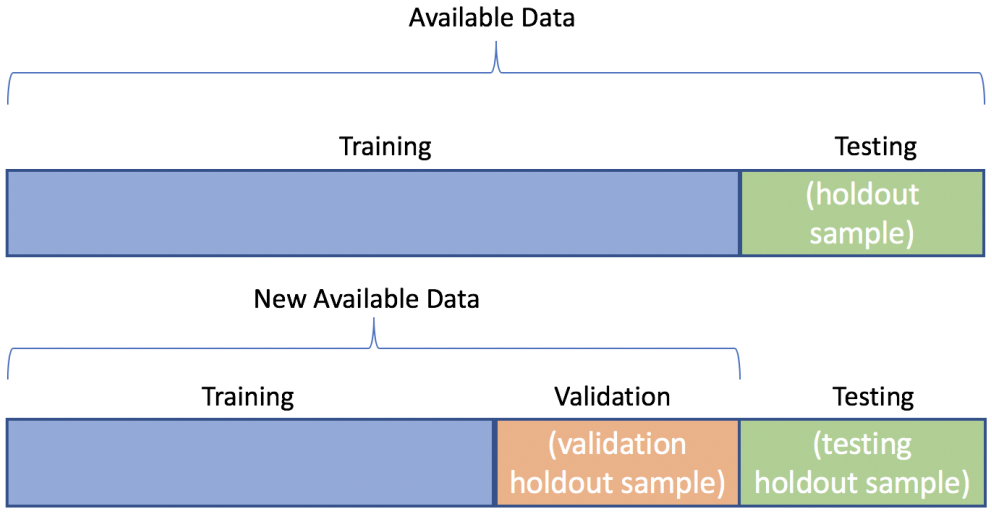
Example reinforcement-learning

- Minecraft is a complex game where you can't describe all relevant scenarios and their solution to train a system
- OpenAI trained a reinforcement learning based model
  - The model was pre-trained on videos using super-vised learning
- Similar principles can be used for self-driving cars

https://openai.com/index/vpt/

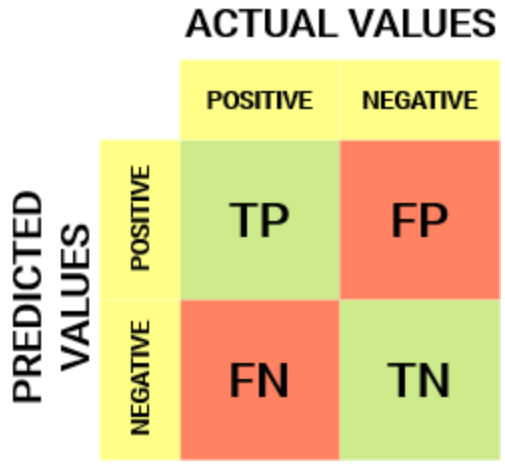
ML concepts

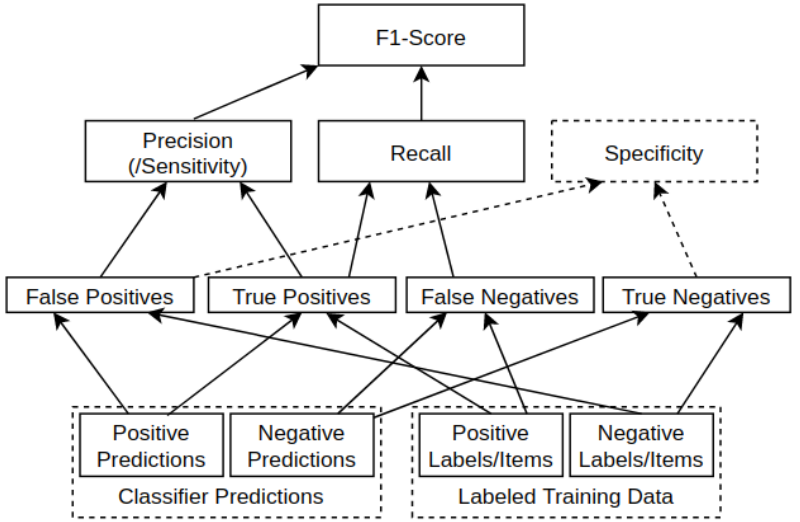
- Data is at the center of all ML
- What data?
  - Train, validation, test
- Quality control of the data
  - How good is my groundtruth?
  - Any missing data?
  - Outliers?
- For the final system the data is just as, if not more, important as the code



Classification metrics

- **Accuracy** : A = (TP + TN) / (TP + TN + FP + FN)
- **Precision** : P = TP / (TP + FP)
  - How many of the found elements are correct
- **Recall** : R = TP / (TP + FN)
  - How many of the correct elements have been found
- Very hard to optimize for both (but easy to optimize for one. 100% recall, put all elements in same cathegory)
- **F1-score** : 2 * (P * R) / (P + R)
  - Best F1-score = 2 * (1 * 1) / (1 + 1) = 1



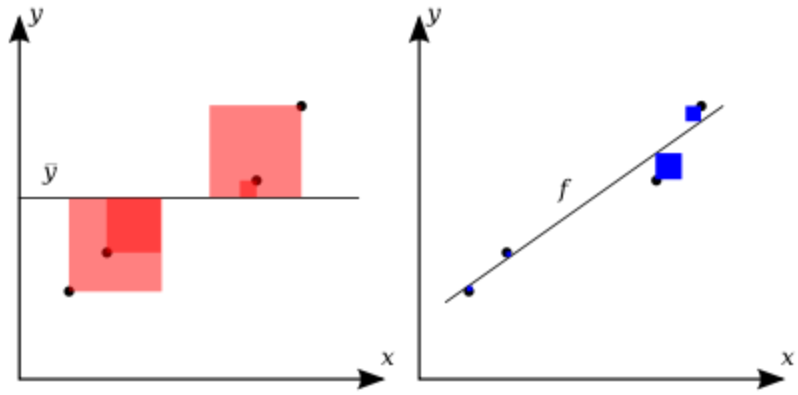


https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

Regression metrics

- **Mean error** : Σ (yi – ŷi) / n
- **Mean absolute error (MAE)** : Σ |yi – ŷi| / n
- **Mean squared error (MSE)** : Σ (yi – ŷi)2 / n

- **Mean relative error** : Σ ((yi – ŷi) / yi) / n
- **Mean absolute relative error** : Σ (|yi – ŷi| / yi) / n
- **R2** (Coefficient of determination)
- Etc.



https://en.wikipedia.org/wiki/Coefficient_of_determination

Generative AI metrics

- Each domain has specific metrics, even more in generative AI
- LLMs have scores like BLEU, or specific Benchmarks
  - Logic reasoning, translations, summarization etc.
- Image diffusion models have Fidelity metrics or diversity metrics
  - They are even harder to evaluate with neutral metrics
- This is why the "Human evaluation" is often used in that domain
- Evaluating your system with clearly defined metrics is key!

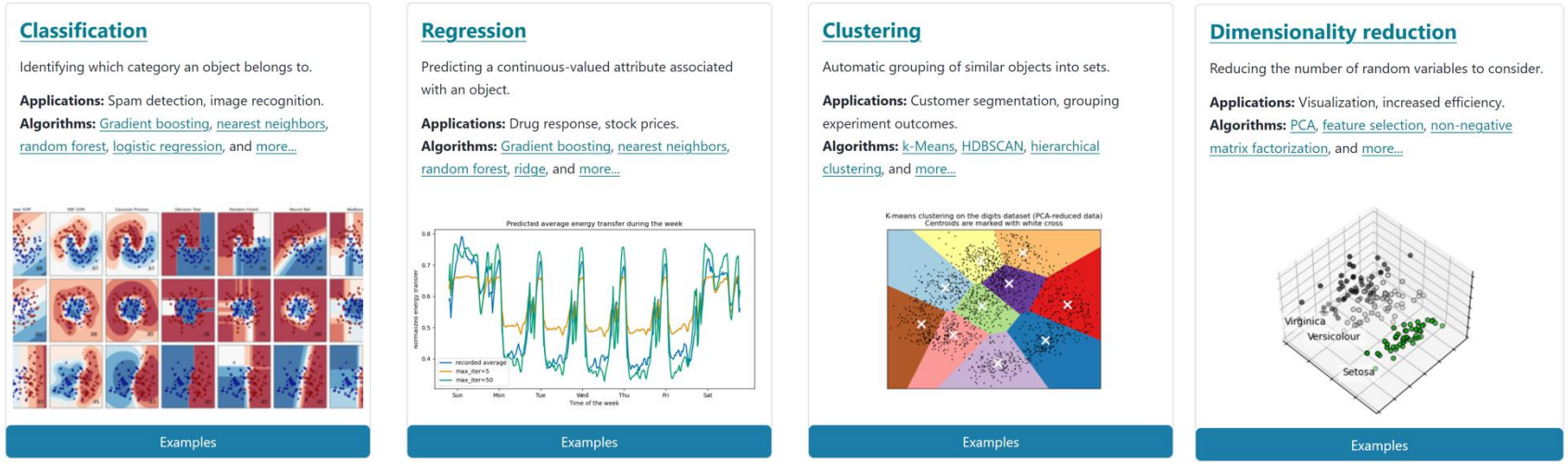
Machine learning

- Once we know what we want to do and how we evaluate the results, we need to implement the project
- The described machine learning methods can be performed with various methods, libraries and frameworks

- Classical machine learning
  - Based on simple, fast models, using statistics and similar methods
- Deep learning
  - Using large models, often requiring GPUs to run efficiently

Classic approaches

- Classic approaches existed for a long time
- They are usually fast, easy to use and well understood
- Libraries like scikit-learn allow you to use a variety of models



Scikit-learn example

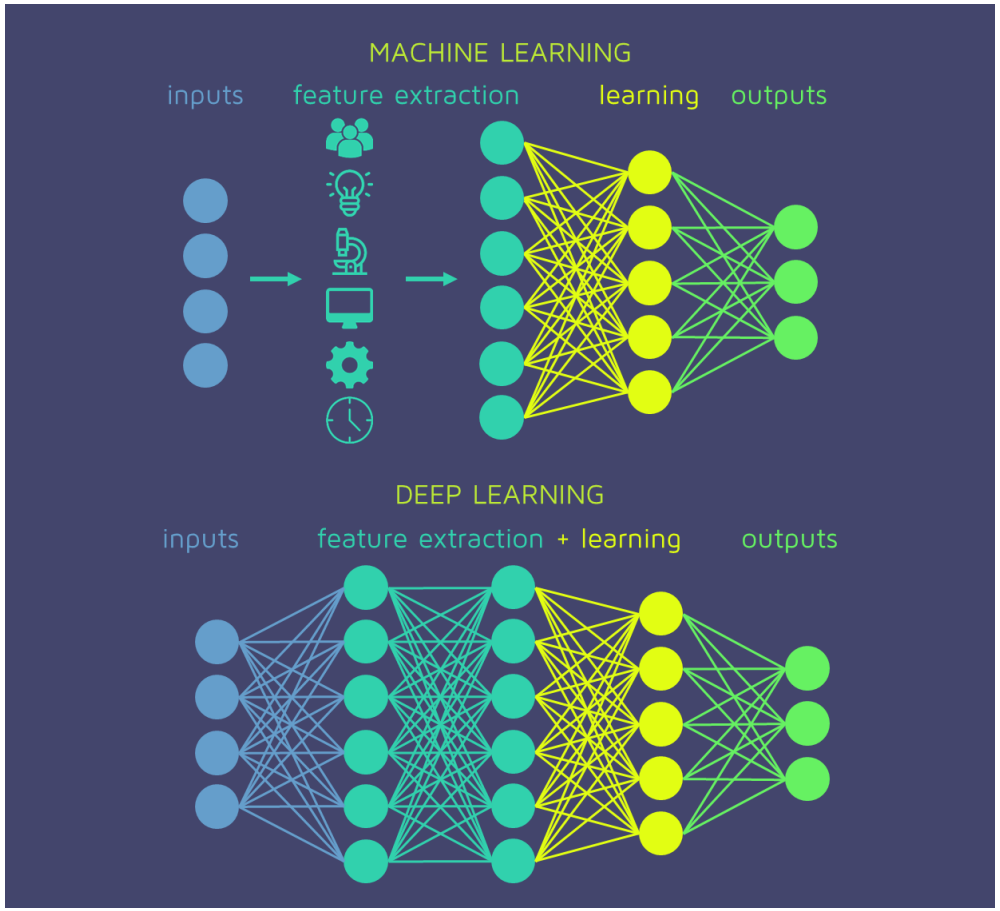
- Using scikit-learn based models is very simple
- Most models in scikit-learn use very little resources and share the same API
- While their power is limited, they are often very good, at the very least as baselines

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target labels

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create a KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)

# Train the classifier
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 100.00%
```

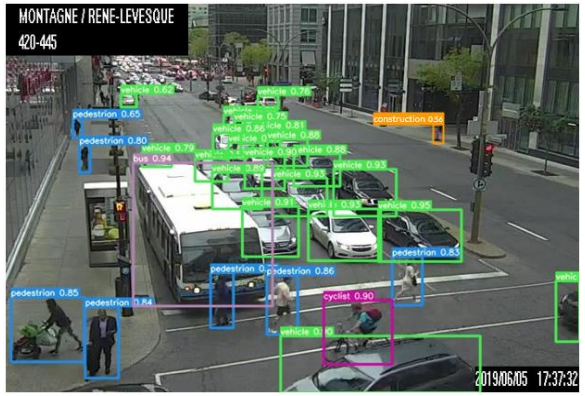
Deep learning approaches

- Deep learning is machine learning, but delegates feature extraction to the model
  - And adds a lot of complexity
- Various methods and frameworks exist to use Deep Learning
- GPUs are the main hardware on which those models run
  - The increased complexity requires much more hardware resources than classic models



Deep learning types

- Deep learning applies to all the types of machine learning seen previously
  - In particular non-supervised learning or semi-supervised learning and generative AI mostly using deep learning approaches



A rabbit detective sitting on a park bench and reading a newspaper in a victorian setting

Deep learning ecosystem

- While for classic machine learning, libraries like scikit-learn are a good reference, deep learning is much more varied
- Low-level libraries like Pytorch, Tensorflow or Jax provide low level primitives to build custom models
- High level libraries like Yolo, llamacpp, whisper and others provide specialized, easy to use models
- Compared to classic machine learning, the speed of new releases is extreme

Practical examples

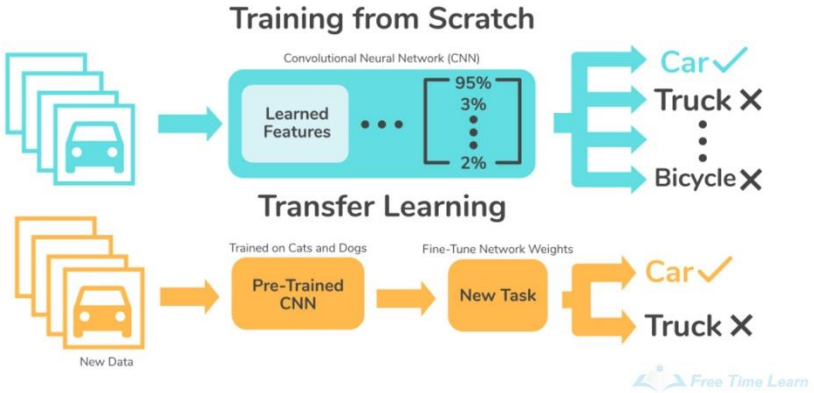
- Many high-level libraries are very easy to use
  - The question becomes, how do I validate the results and scale?

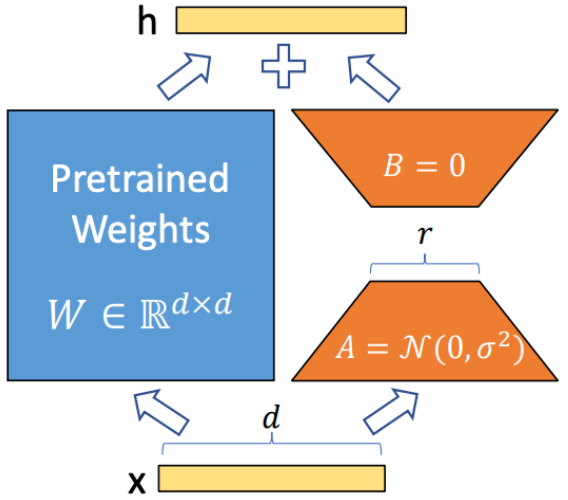
```
import whisper
model = whisper.load_model("base")
result = model.transcribe("audio.mp3")
print(result["text"])
```

```
from transformers import pipeline
generator = pipeline('text-generation', model='gpt2')
generator("Once upon a time,", max_length=30, num_return_sequences=1)
```

Adapting models

- Full training
  - Train a model from scratch with your data
- Pre-trained model
  - Using a model pre-trained on your task. Eg. Yolo to detect people
- Zero shot
  - Use a pre-trained model on a problem it has never seen. E.g. Find a zebra in a video, without having trained on zebras
- Transfer learning
  - Use a pre-trained model and fine-tune it on your problem
- Loras
  - A type of light weight fine-tuning of large models





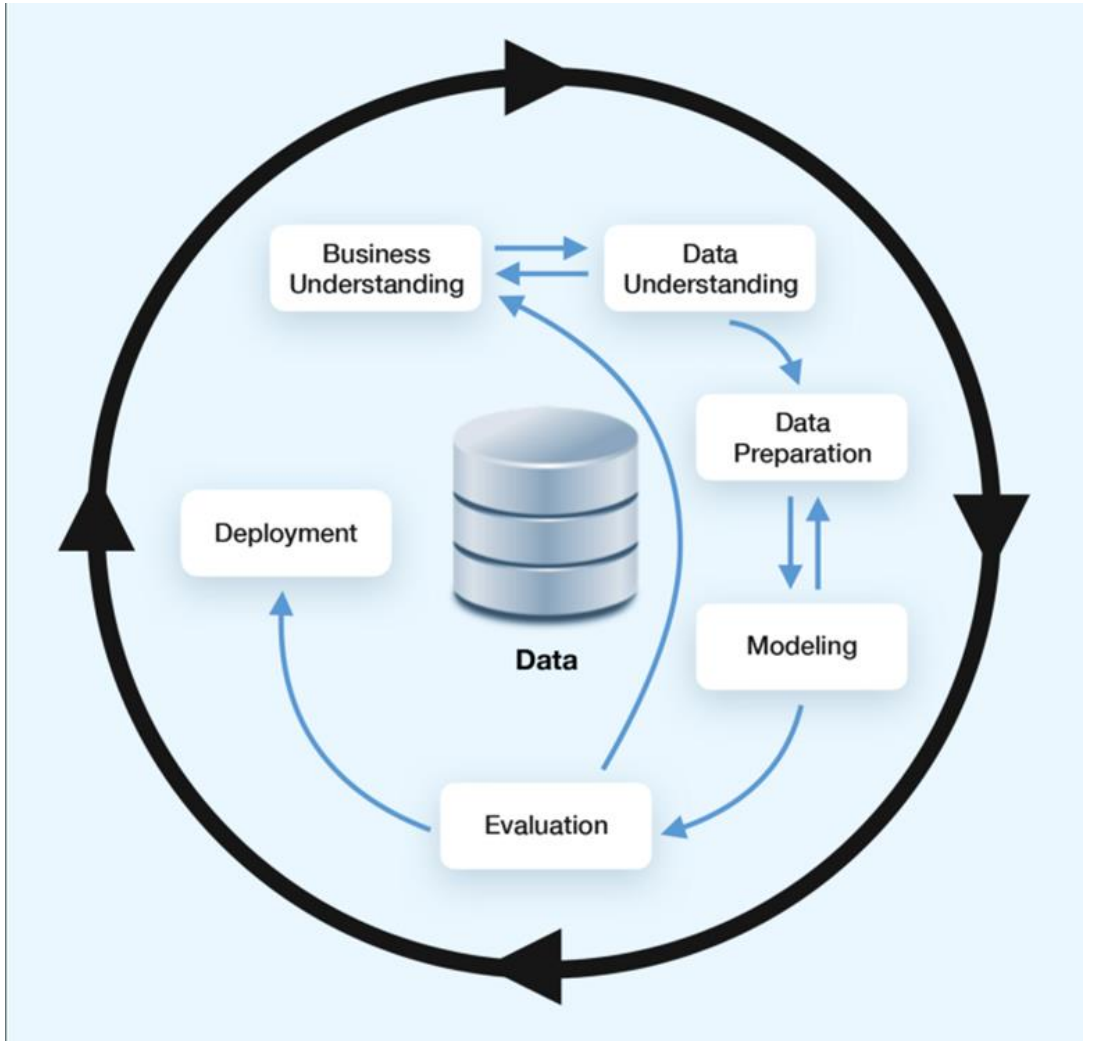
Machine learning conclusion

- Machine learning is a vast domain of applications and methods
- When doing machine learning we need to:
  - Define the task correctly
  - Define the metrics used to evaluate the task
  - Select a method of solving the problem
    - Training, zero-shot, transfer learning etc.
  - Select data to train and evaluate
  - Find the correct way to actually bring it all to production
- Selecting the right solutions is key to a successful project

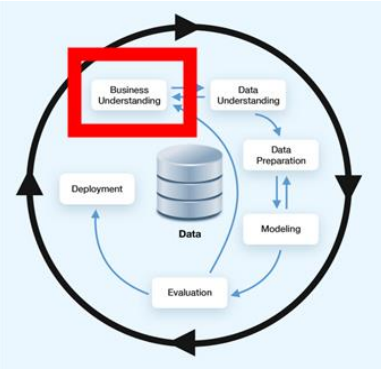
Project management

Classic data analysis project management

- The Cross-Industry Standard for Data Mining (CRISP-DM) is a standard model for classic data analysis projects
- Circular approach developed by IBM
  - Business Understanding
  - Data Understanding
  - Data Preparation
  - Modeling
  - Evaluation
  - Deployment

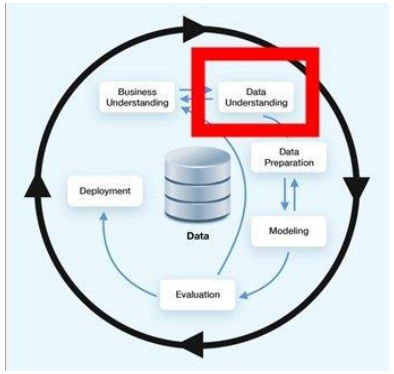


Business Understanding



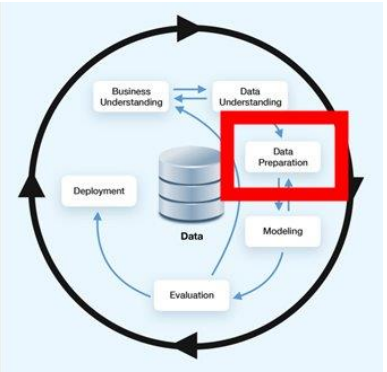
- **Define Business Objectives**: Understand from the client's perspective what we want to achieve and formulate success criteria
- **Situation Assessment**: Define the available resources, project requirements, risks, and perform a cost/benefit analysis
- **Data Exploration Objectives**: In addition to defining business success, determine what success means from a data analysis perspective
- **Project Plan**: Choose technologies and provide a detailed plan for each stage of the project.

Data Understanding



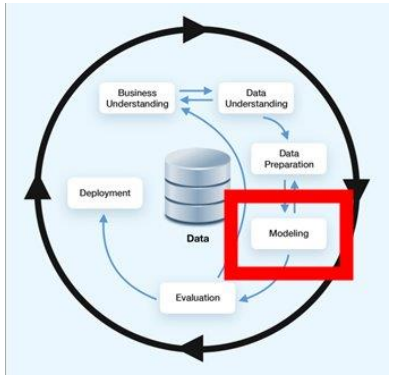
- **Initial Data Collection**: Acquire the necessary data and load it into analysis tools
- **Data Description**: Examine the data and document its high-level properties (data format, number of entries, columns, etc.)
- **Data Exploration**: Perform a deeper analysis of the data. Query, visualize, identify correlations, etc.
- **Data Quality Verification**: How clean is the data? Document any quality issues.

Data Preparation



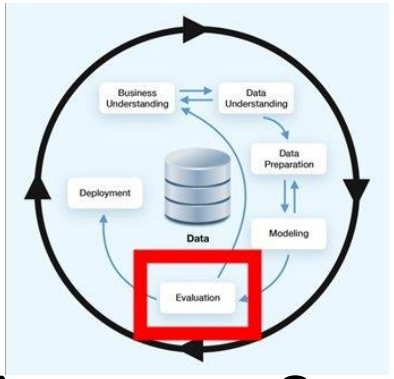
- **Data Selection:** Determine which datasets will be used and why (and why not for the negatives)
- **Data Cleaning:** One of the largest tasks (otherwise, garbage-in, garbage-out). Correct, impute, or delete bad data
- **Constructing New Data:** Derive new attributes that could be useful. Example: BMI from a person's height and weight
- **Data Integration:** Create a new dataset by integrating data from multiple sources
- **Data Formatting:** Reformat data if necessary. For example: String to numeric, normalization, etc.

Modeling



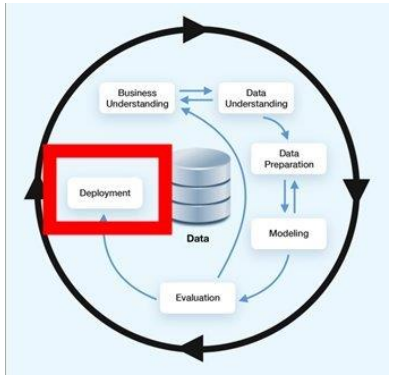
- **Selection of Modeling Techniques**: Determine which algorithms to use (regression, neural networks, etc.)
- **Generation of a Test Design**: Depending on the approach, split the data into "training", "test", and "validation" sets. Define model quality criteria.
- **Model Creation**: Build the model itself. Sometimes it comes down to: *reg = LinearRegression().fit(X, y)*.
- **Model Evaluation**: Often multiple models are compared, and we need to interpret the results based on our domain understanding, defined success criteria, and test design.

Evaluation



- **Evaluation of Results**: Do our models meet the requirements? Which model(s) should be chosen for the final product?
- **Review Process**: Review the work. Did we miss anything? Were all steps executed correctly? Summarize this and correct if necessary.
- **Determination of Next Steps**: Based on the previous tasks, determine if our models can be deployed or if we need to iterate further.

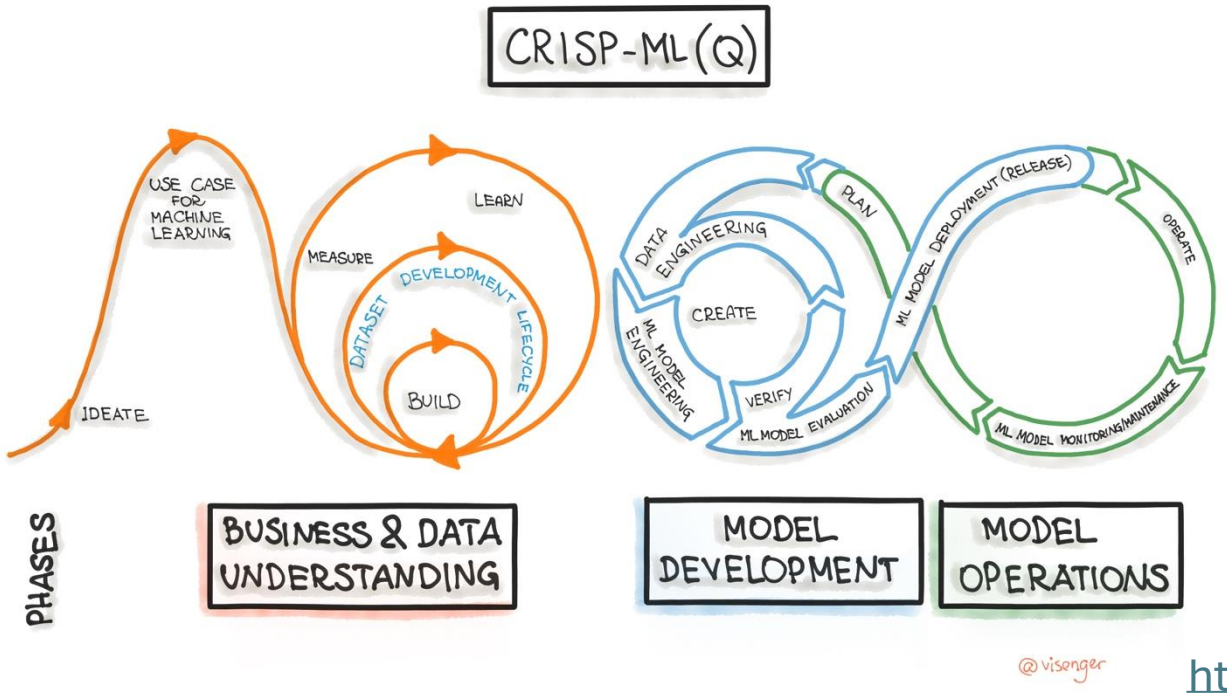
Deployment



- **Deployment Planning**: Create and document a plan for deploying a model
- **Monitoring and Maintenance Planning**: Set up a system to monitor the deployed model to prevent issues during the operational phase
- **Production of a Final Report**: Create a final report that documents all the results
- **Execution of a Final Project Review**: Retrospective on what went well or poorly during the project and how to improve in the future.

CRISP-DM conclusion

- CRISP-DM is a good guide to follow in a big data analysis project
- It pre-dates MLOps and modern machine learning
- Evolutions of CRISP-DM exist, pushing the iterative nature even further



https://ml-ops.org/content/crisp-ml

ML and data analysis conclusion

- Data analysis is at the core of many projects
- Most data analysis projects produce some kind of machine learning model
- Machine learning is a large field with many methods and frameworks
  - Data, training or validation, is key for all of them
- Data quality, model metrics and scaling are unique issues facing those projects

Introduction to MLOps

Bertil Chapuis

Beat Wolf

Machine learning in operations

- **DevOps, DataOps, MLOps:** Are these just buzz words?
- They illustrates a **shift of paradigm** in how teams build, deploy, and manage technology.
- Dev, Data, and ML each come with **unique challenges** once in operation.
  - **Specialization:** Domain-specific expertise.
  - **Automation:** Streamlining repetitive tasks to enhance efficiency
  - **Collaboration:** Cross-functional teamwork to break down silos.
  - **Scalability:** Building systems that adapt to the demand.
  - Etc.

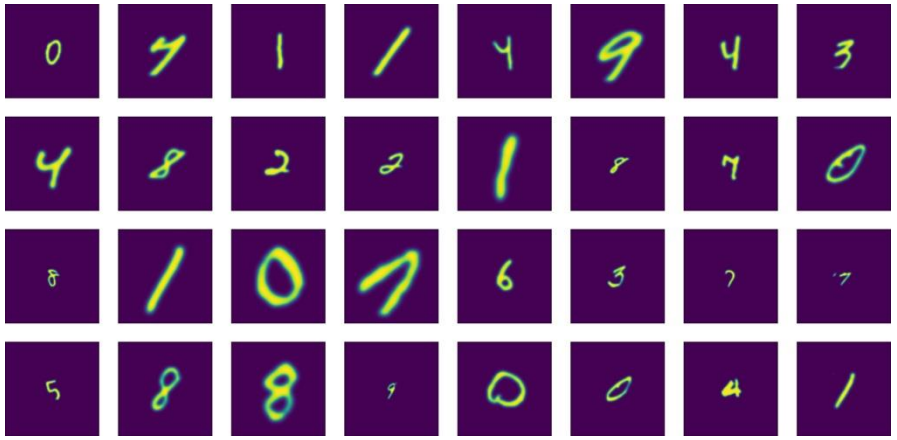
Traditional software vs machine learning

- The code generates outputs from data
  - Testing and versioning applies only to the code
- The code and the data produce ML models
  - Testing and versioning applies to the code, data, and ML models

- This leads to complex problems:
  - How to version a large dataset?
  - How to know if the data is good?
  - How to deploy a very large ML model?
  - How to monitor and debug a ML model in production?
  - Etc.

Machine learning in academia

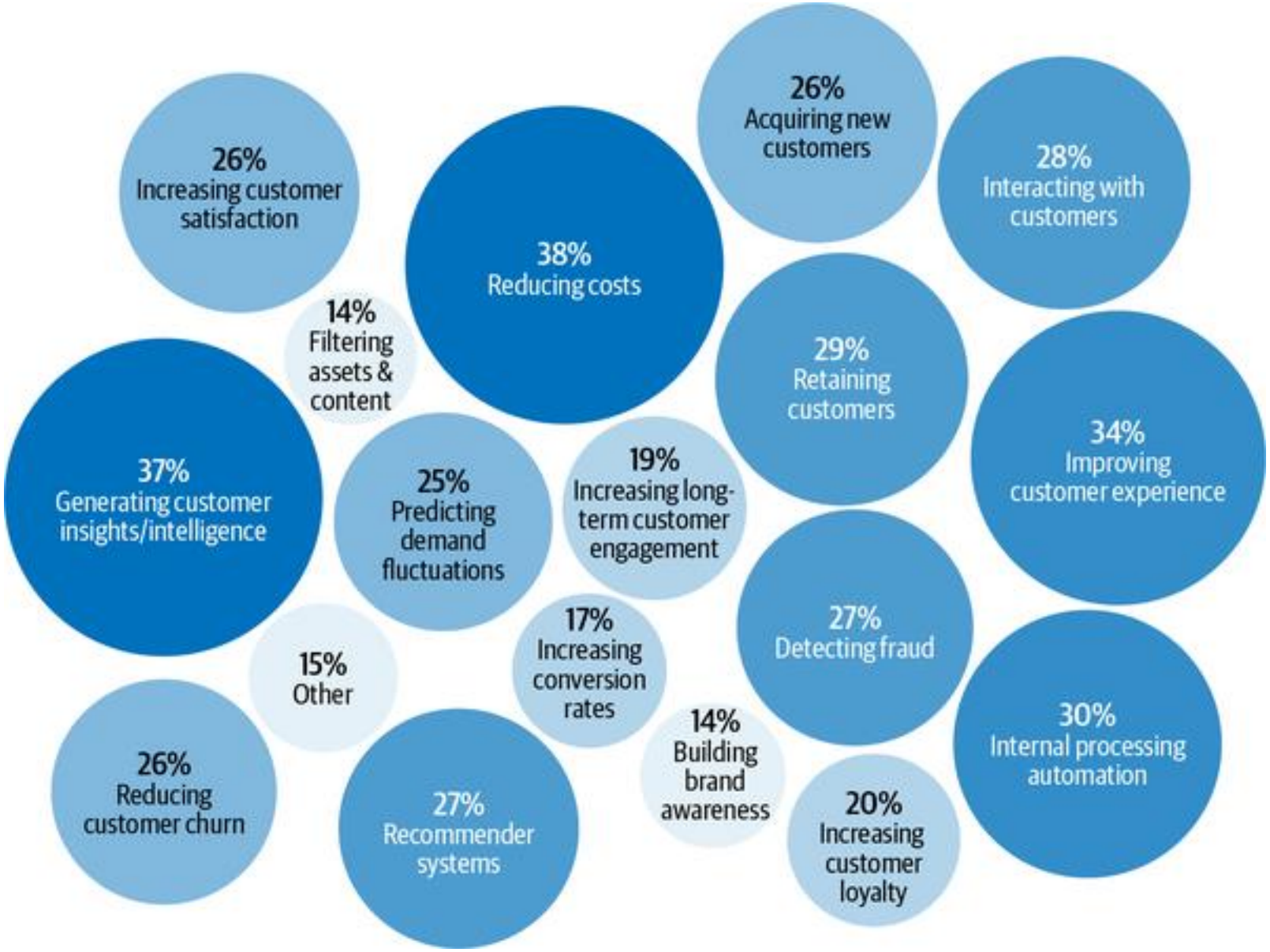
- Most of the time:
  - The problem is well known and described
  - The dataset is ready and static
  - Benchmarks and the baselines are available
  - The scientific work is done in a notebook or a collection of scripts
  - The algorithms and the resulting ML model are described in a publication
  - The impact of a failure is low
  - The data scientist can move on to the next problem
- Things are vastly different in the industry



MNIST dataset

Machine learning in the industry

- A **machine learning system** is more than a machine learning model.
- It includes:
  - **Business requirements** (objectives of the project)
  - **Data pipelines** (data collection, preparation, etc.)
  - **Algorithms** (neural networks, decision trees, llm, etc.)
  - **Infrastructure** (hardware and cloud services for training and inference)
  - **Development and Monitoring tools** (to train, monitor, and update models)
  - **User interfaces** (how people interact with the system)
  - Etc.



Source: https://learning.oreilly.com/library/view/designing-machine-learning/

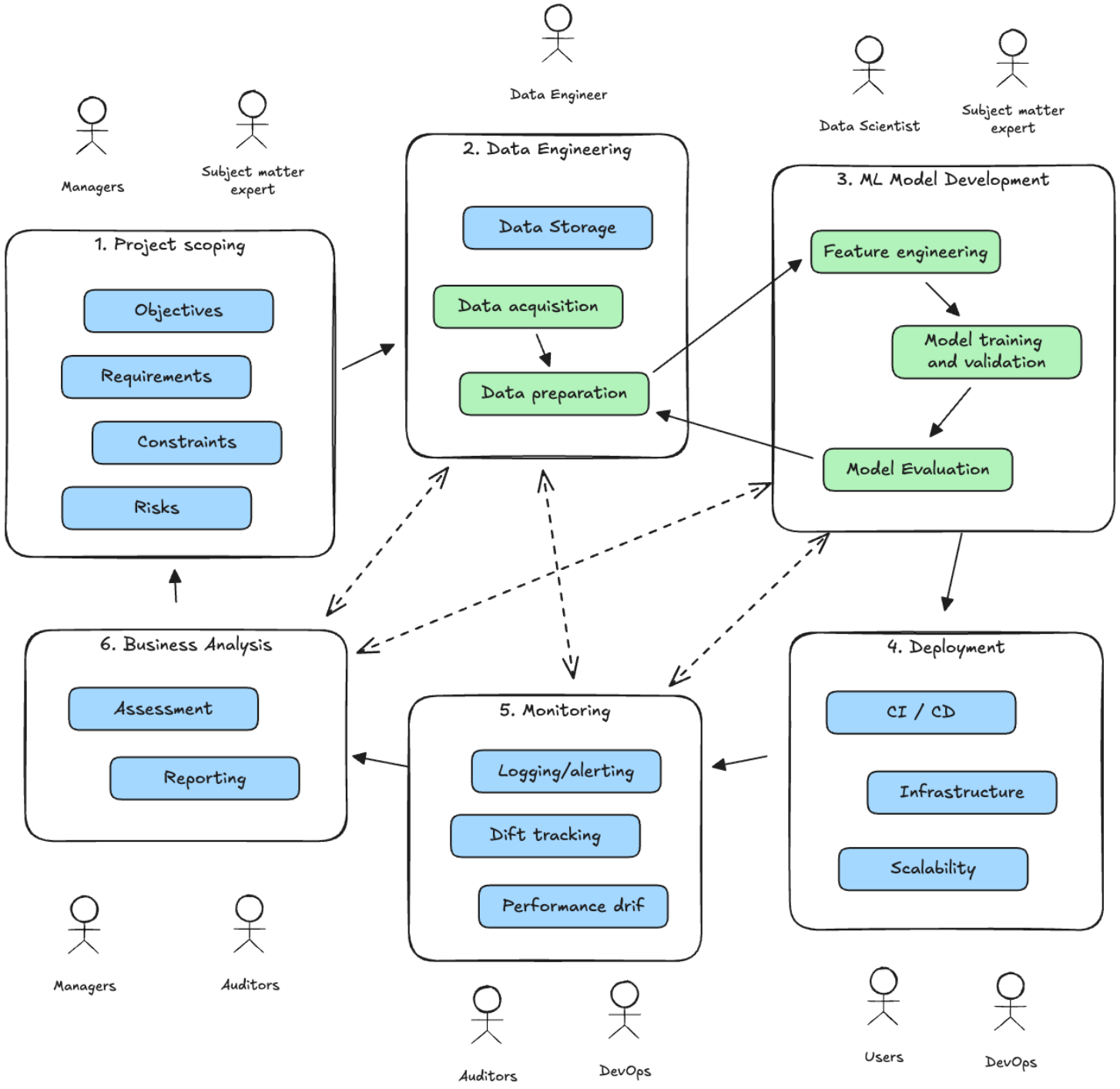
Life cycle of a machine learning system (solo)

- **Project Scoping:** Choose a metric.
- **Data Preparation:** Collect and label data.
- **Feature Engineering:** Create relevant features for the model.
- **Model Training :** Train the model with current data.
- **Model Evaluation:**
  - Identify and fix data issues (e.g., incorrect labels).
  - Address data imbalance (e.g., too many negative labels).
  - Update with recent data.
- **Model Deployment:** Deploy the model in production.
- **Analysis:** Monitor business impact and update objective if needed.

- Repeat as necessary.

Life cycle of a machine learning system (team)

In larger teams, things get more complicated, and the people involved need to collaborate.



1. Project scoping

- **Defining the context and business goals:**
  - Describe the problem and expected results
  - Identify the stakeholders
  - Identify the requirements
- **Defining the machine learning problem:**
  - Input
  - Desired output
  - Type of machine learning problem (regression, classification, etc.)
  - Objective function(s)
- **Estimate and allocate resources (data, compute, humans, etc.)**
- **Assess and mitigate the risks**
- Etc.

Stakeholders of a machine learning system

Machine learning systems usually require **cross-functional teams**:

- Users
- Business expert
- Subject matter experts
- Data scientists
- Data engineers
- DevOps engineers
- Software engineers
- Auditors
- System architects
- Etc.

One person can hold several roles, and the **full-stack data scientist** is becoming a thing. However, this only works if **tooling remains simple**.

Requirements of a machine learing system

- Functional: *The model should be able to predict house prices based on features such as location, size, number of bedrooms, and age of the property.*
- Non-functional: The model should use no more than 256 GB of RAM during inference to ensure it can run on resource-constrained devices.
  - Reliability
  - Scalability and performance
  - Maintainability
  - Adaptability
  - Data privacy
  - Etc.

Requirements in machine learning are harder to enforce.

Example: Wrong results in a translation app may remain unnoticed.

Type of machine learning problem

- You work for a leading e-commerce platform as a Data Scientist.
- Your boss notices that a competitor handles customer support much faster and asks you to address this problem.
- Your customers are frustrated by the long resolution time and say that they had to exchange with multiple person before having a satisfying answer.
- **Where does your job start?**
- **What’s your next move?**

Type of machine learning problem

- **Bottleneck:** Misrouting of customer queries to incorrect support teams leads to increased response time and customer dissatisfaction.
- **ML Problem Type:** Multi-class Classification
- **Input:** Customer support query text ("I need help with my order”, "My payment didn't go through”, etc.)
- **Output:** Correct department (Billing, Product Support, Order Status, Technical Assistance, Delivery, Procurement, etc.)
- **Objective Function:** Minimize misclassification of department routing to improve response time.

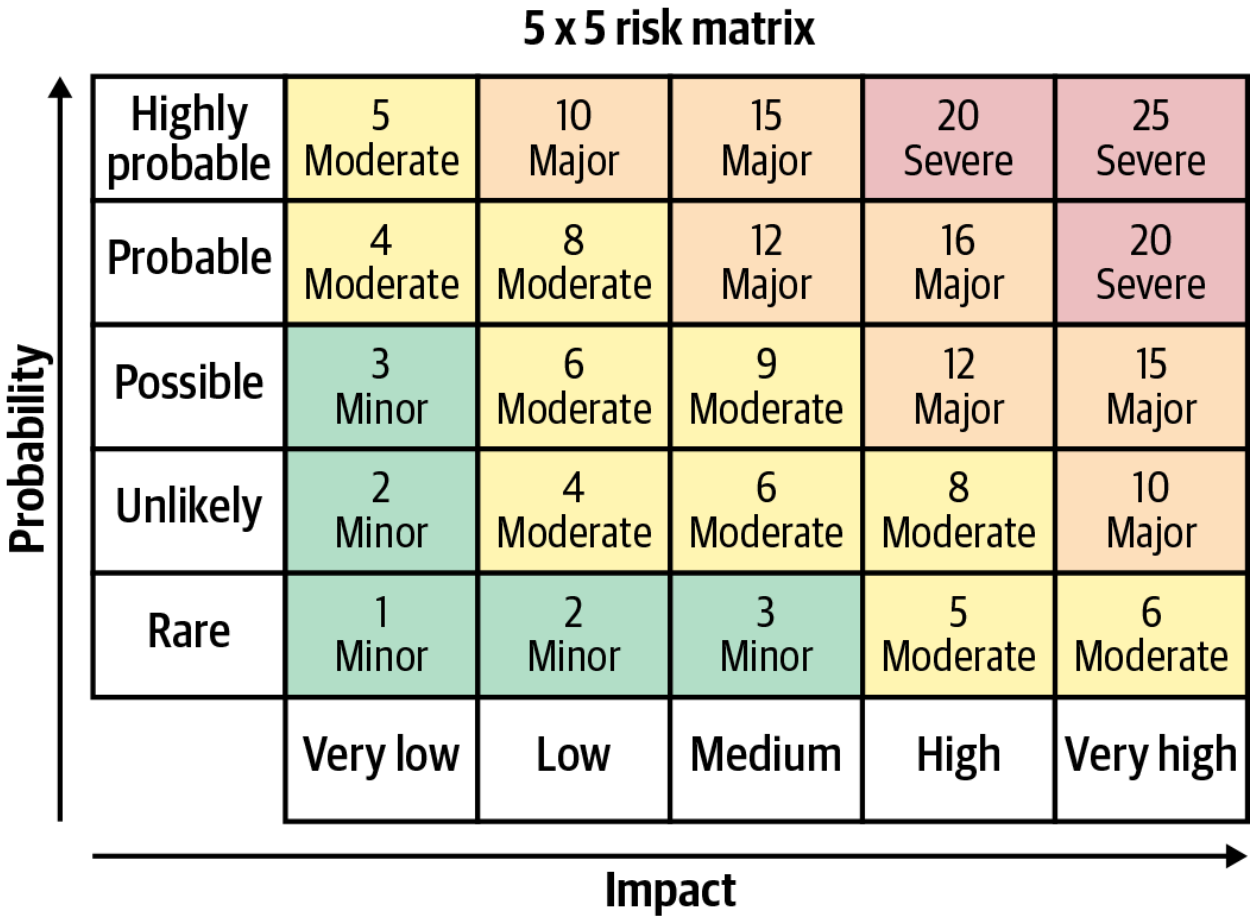
Estimate and allocate resources

- **Data Collection & Pre-processing** (6 weeks: data engineer, ML engineer, 2x interns)
  - Gather historical customer queries
  - Annotate the data with the correct departments
  - Clean and preprocess the data (normalization, tokenization, etc.)
- **Baseline Model Development** (2 weeks: ML engineer)
  - Split the dataset (train, test, validation)
  - Choose a model for text classification (BERT)
  - Fine-tune the model on the training data
  - Evaluate the model performence
- **Model integration** (2 weeks: ML engineer, software developer)
  - Develop an API to serve model predictions
  - Integrate the API with the customer service system
  - Implement logging to monitor model performance and errors
- **System testing and quality assurance** (1-2 weeks: ML engineer, QA engineer, customer support)
  - Test the end-to-end system to ensure correct query rooting
  - Perform user acceptance testing with a few customer support agents
  - Gather initial feedbacks and fix issues.
- **Deployment and monitoring** (1 week: DevOps engineer, ML engineer)
  - Deploy the model to production
  - Plan for periodic evaluation and updates

Assess and mitigate the risks

- What if the model makes wrong predictions?
- What if the model accuracy decreases over time?
- What if the system gets unavailable?
- What if a key stakeholder leaves the company?
- Who takes the responsibility in case of failure?
- Etc.

Assess and mitigate the risks



Current status:
We already misroute most of customer queries.

Probability: *possible*

Impact: *very low*

What if we don't get a green light?

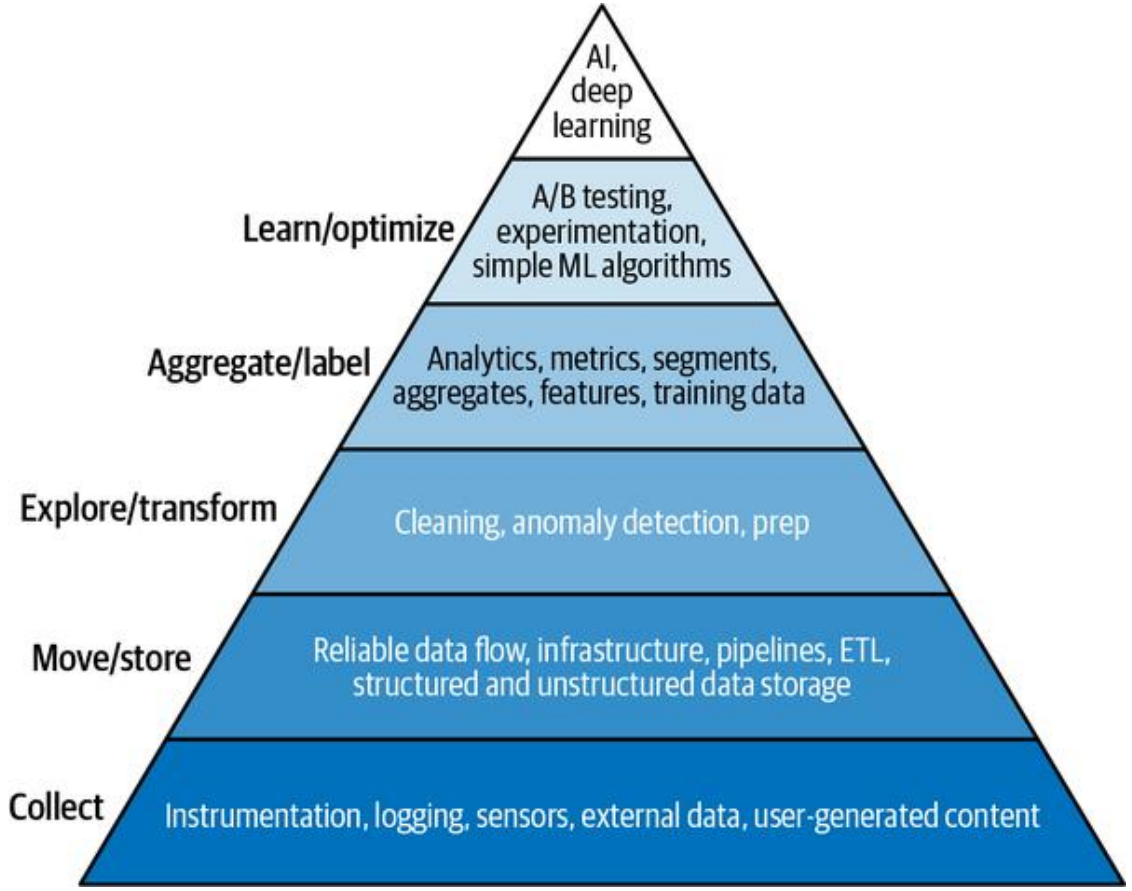
Responsible AI

- **Intentionality**
  - Ensuring that models and systems are developed and function in alignment with their intended purpose.
- **Accountability**
  - Ensuring that there are clear mechanisms and structures in place to hold stakeholders responsible for the outcome of the models and systems.

2. Data engineering

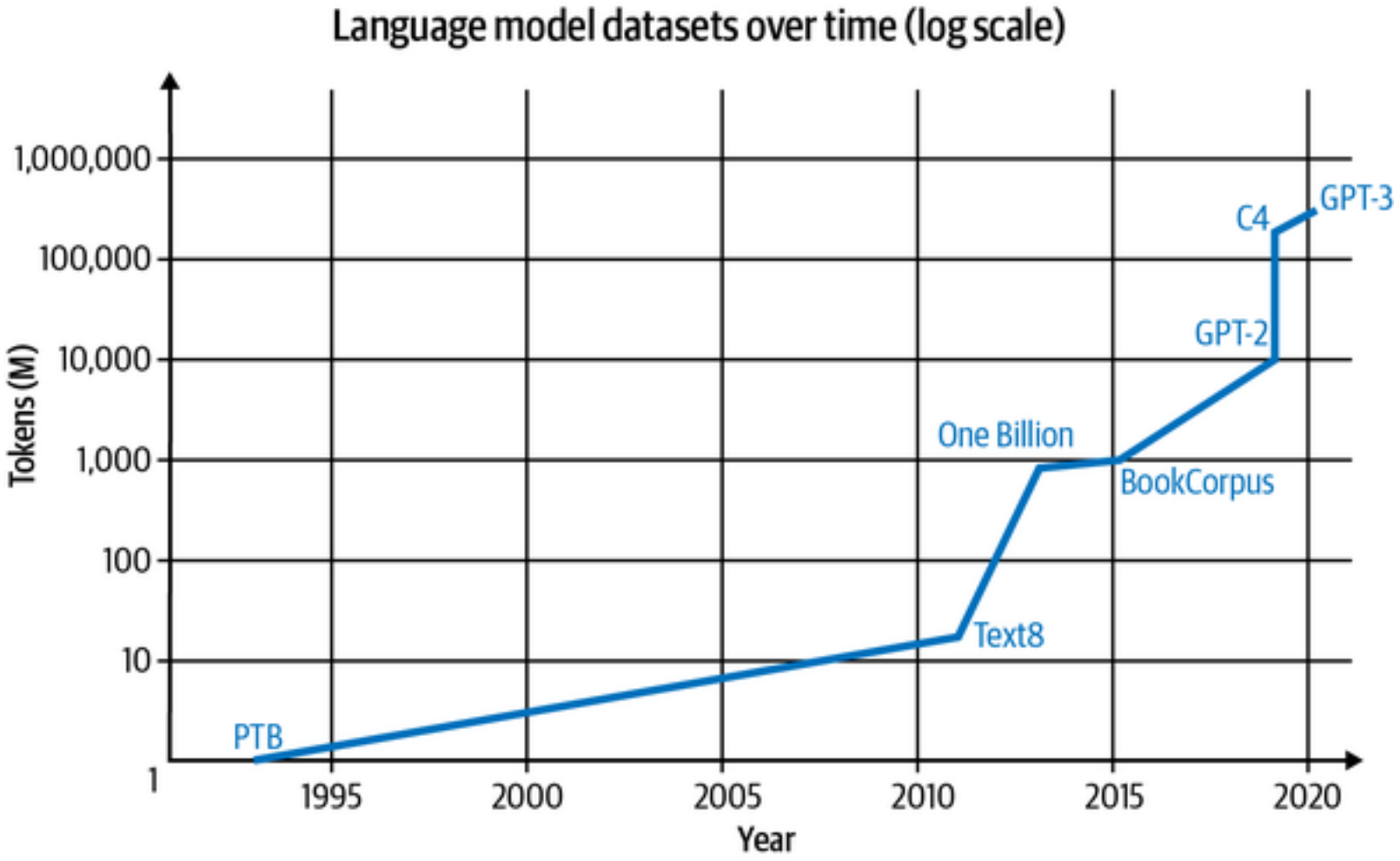
- Data collection:
  - Gather data from various sources.
  - Ensure data relevance and quality.
- Data preparation:
  - Clean and preprocess the data.
  - Handle missing values and outliers.
- Data storage:
  - Organize data for efficient access.
  - Implement data versioning.
- Etc.

Data as a Foundation for machine learning



Source: https://learning.oreilly.com/library/view/designing-machine-learning/

On the importance of data...



Source: https://learning.oreilly.com/library/view/designing-machine-learning/

3. Model development

- **Feature Engineering:**
  - Extract meaningful features from raw data.
  - Use domain knowledge to enhance features.
- **Algorithm Selection:**
  - Choose appropriate ML algorithms for the task.
- **Training and Validation:**
  - Split data into training, validation, and test sets.
  - Optimize model parameters and hyperparameters.
- **Evaluation:**
  - Assess model performance using relevant metrics.
  - Perform error analysis to identify improvement areas.
- Etc.

4. Deployment

- **Model integration:**
  - Embed the model into the production environment.
- **Scalability considerations:**
  - Ensure the system can handle increased load.
- **Latency and performance optimization:**
  - Reduce inference time for a better user experience.
  - Batch processing and streaming
- **Evaluation and comparison:**
  - Compare new model versions with existing ones.
- Etc.

5. Monitoring

- **Performance Tracking:**
  - Monitor key metrics in real-time.
  - Detect model drift and data distribution changes.
- **Feedback Loop:**
  - Incorporate user feedback to improve the model.
- **Model Updating:**
  - Retrain or fine-tune the model with new data.
- **Alerting Mechanisms:**
  - Set up notifications for anomalies or degradations.
- Etc.

6. Business Analysis

- **Impact Assessment:**
  - Evaluate the model's effect on business metrics.
- **Reporting:**
  - Communicate findings to stakeholders.
- **Strategic Decisions:**
  - Decide on scaling, pivoting, or discontinuing projects.
- **ROI Calculation:**
  - Analyze return on investment for the machine learning system.
- Etc.

MLOps Workshop

Simple tooling for the full-stack data scientist: