Project organisation

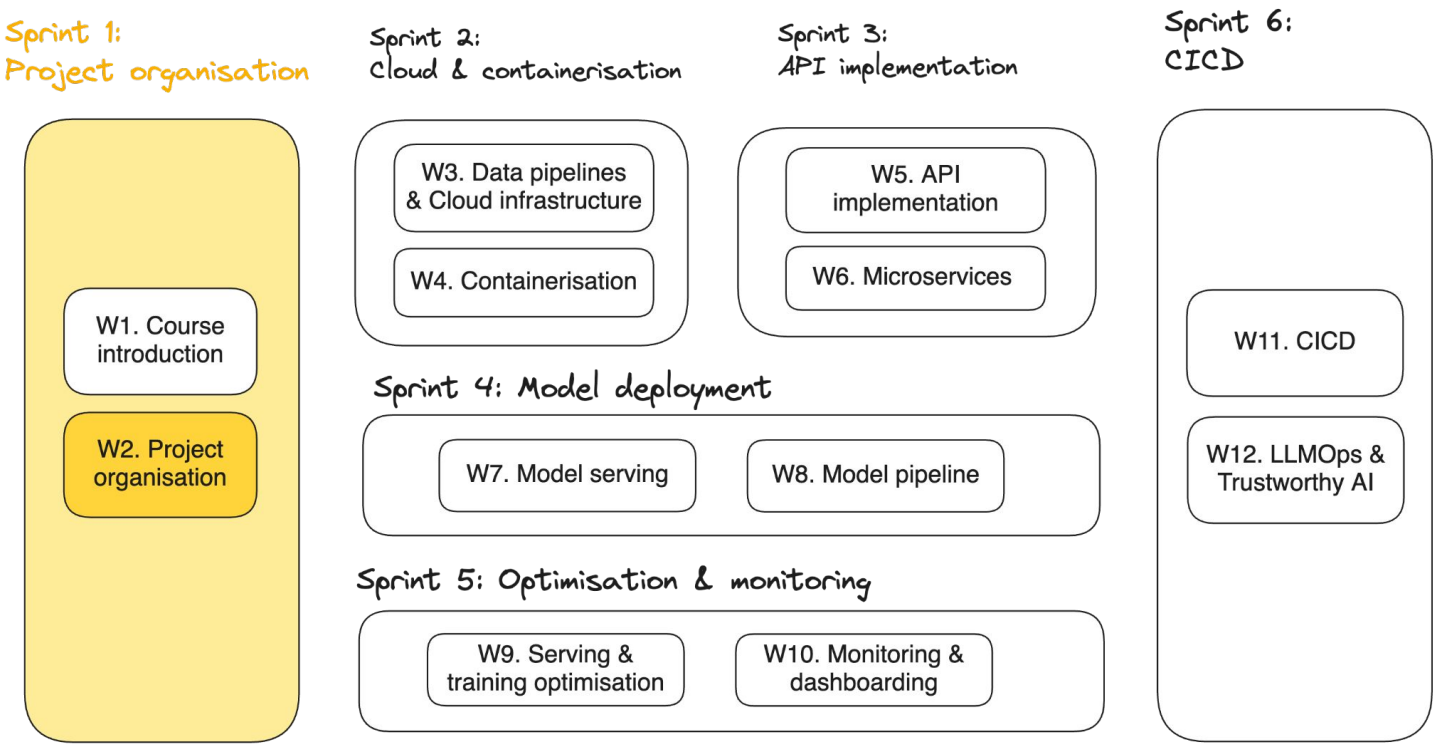
Sprint 1 - Week 2

INFO 9023 - Machine Learning Systems Design

Thomas Vrancken (t.vrancken@uliege.be)
Matthias Pirlet (matthias.pirlet@uliege.be)

2025 Spring

Status on our overall course roadmap



It’s hard to organise an IT project…

Agenda

What will we talk about today

Lecture

- Agile way of working
- Data sources
- Code versioning & conventions

Demo

- Git code versioning

Project objective for sprint 1

Week	Work package	Requirement
W01	Pick a **team** ● Try to mix skills and experience ● If you didn't find one let one of the teachers know and we'll allocate you to one	Required
W02	Select a **use case** ● Previous course ● Kaggle Datasets ● … Make sure to pick a use case where **data is available**. Ideally pick something with interesting data and a real world application.	Required
W02	**Define** your use case. Fill in a ML Canvas template page (You can skip the *Inference* part as we will tackle that in a later sprint.)	Required
W02	Find a **cool name** for your project ✨	Required
W02	Setup **communication channel** (Discord, trello)	Required
W02	Setup a **code** versioning **repository** ● We recommend Github as we will cover Github Actions during this course	Required
W02	Submit your project by sending a filled in project card to the teaching staff with basic information about your project. We might give you some feedback and ask for parts to be changed.	Required

Project deliverables

Deliverables

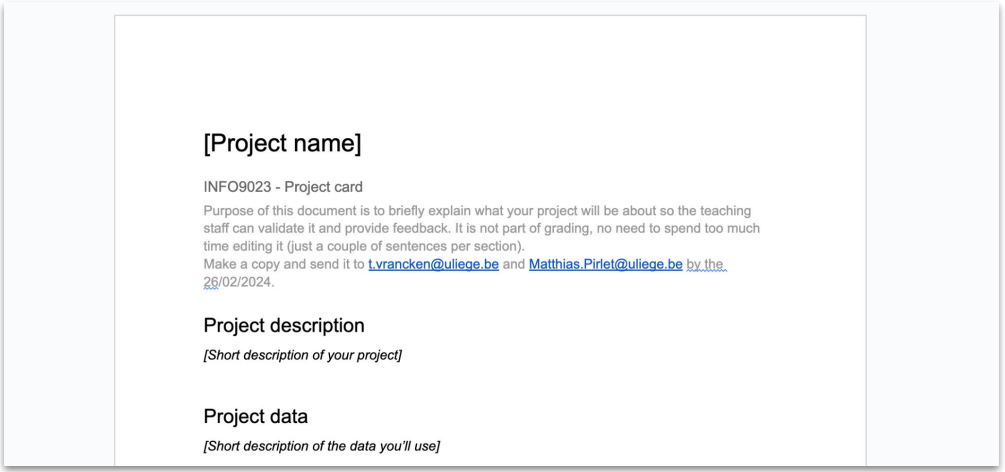
- **3 Milestone (MS) meetings**:
  - MS 1: Present work from *sprint 1 & 2*
    - Present your general use case (BMC), the data preparation and the result from your model experimentation.
  - MS 2: Present work from *sprint 3 & 4*
    - Present your architecture for model deployment and automated training.
  - MS 3: Overall presentation
    - Whole project, inc. sprint 5
- Each MS presentation will be accompanied with a *code submission on Github*. Teaching staff will not go over all codes but key information can be provided in README + check that everything is well implemented.

No handovers outside of the MS (presentation & code submission). You won't have to submit something every sprint. The work packages per sprints are there to guide you, but you're free to implement it at your pace. Teaching staff is there to provide support every week.

Project deliverables

⚠️ Submit your **team** and **project** by filling in this project card template (make a copy) and sending it to the teaching staff.

- Purpose is for the teaching staff to validate the project ideas and potentially provide feedback.
- Fill it in before next lesson.



[Project name]

INFO9023 - Project card

Purpose of this document is to briefly explain what your project will be about so the teaching staff can validate it and provide feedback. It is not part of grading, no need to spend too much time editing it (just a couple of sentences per section).
Make a copy and send it to t.vrancken@uliege.be and Matthias.Pirlet@uliege.be by the 26/02/2024.

Project description

[Short description of your project]

Project data

[Short description of the data you'll use]

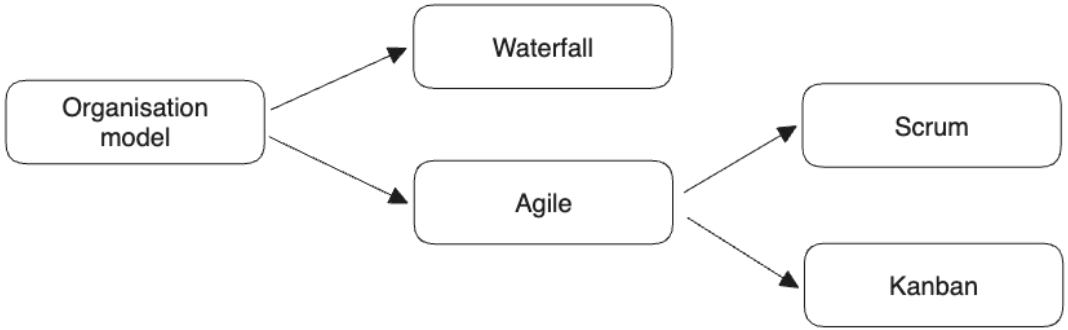
LIÈGE université

Agile way of working

Why are we talking about organisation models?

1. Projects more often fail due to poor organisation rather than lack of technical knowledge
2. You won’t always have a “Project Manager” to organise your project
3. … and if you do, you still have to work Agile, so better know it

Different models for organising an IT project

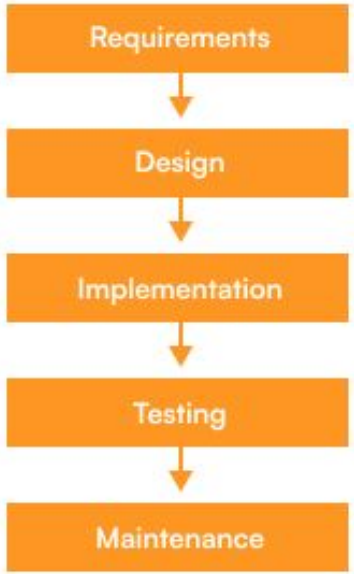


Waterfall model

Models for organising IT projects and teams

Organise a project as a **waterfall**, flowing down **sequentially** through 5 phases. It is **rigid**, **structured** and **linear** (it is not Agile).

Decision maker team sets a detailed plan for the whole project implementation, with limited flexibility.



Waterfall methodology

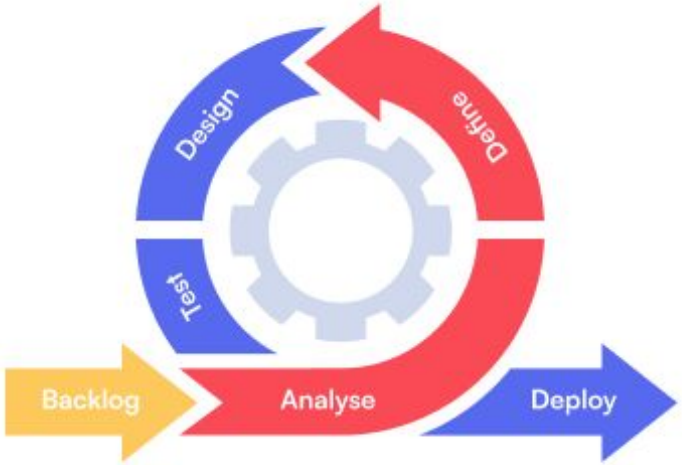
Agile model

Models for organising IT projects and teams

Iterative approach to delivering a project, which focuses on **continuous releases** that incorporate **customer feedback**.

Ability to **adjust** during each iteration promotes **velocity** and **adaptability**.

This approach is different from a linear, waterfall project management approach, which follows a set path with limited deviation.



Agile methodology

Waterfall vs Agile

Criteria	Agile	Waterfall
Approach	Iterative and Incremental	Sequential
Flexibility	Highly flexible and adaptable	Less flexible
Planning	~~Minimal~~ Less planning is enough	Detailed planning required
Customer Involvement	High level of customer involvement	Low level of customer involvement
Risk management	Continuous risk management	Risk management at the beginning of the project
Documentation	~~Minimal~~ Less documentation required	Extensive documentation required
Time and cost	Challenging to estimate time and cost	Easy to estimate time and cost

https://testsigma.com/blog/difference-between-agile-and-waterfall/

Agile:
Scrum vs Kanban

Kanban methodology

In Japanese, kanban literally translates to "visual signal."

- Originally comes from *lean manufacturing* in Japan (at Toyota)
- Simple and clean application of Agile
- Matches the amount of Work in Progress (WIP) to the team's capacity - *Just In Time (JIT)*
- Minimal set of rules - easy to pick up for new software engineering teams

Advantages

+ Planning flexibility
  - Once done with an item, pick up the next on the backlog
  - Product owner can update the backlog without disrupting the team
+ Short time cycles
  - Time for a unit of work to cycle through the whole process
+ Fewer bottlenecks
+ Visual metrics, transparency

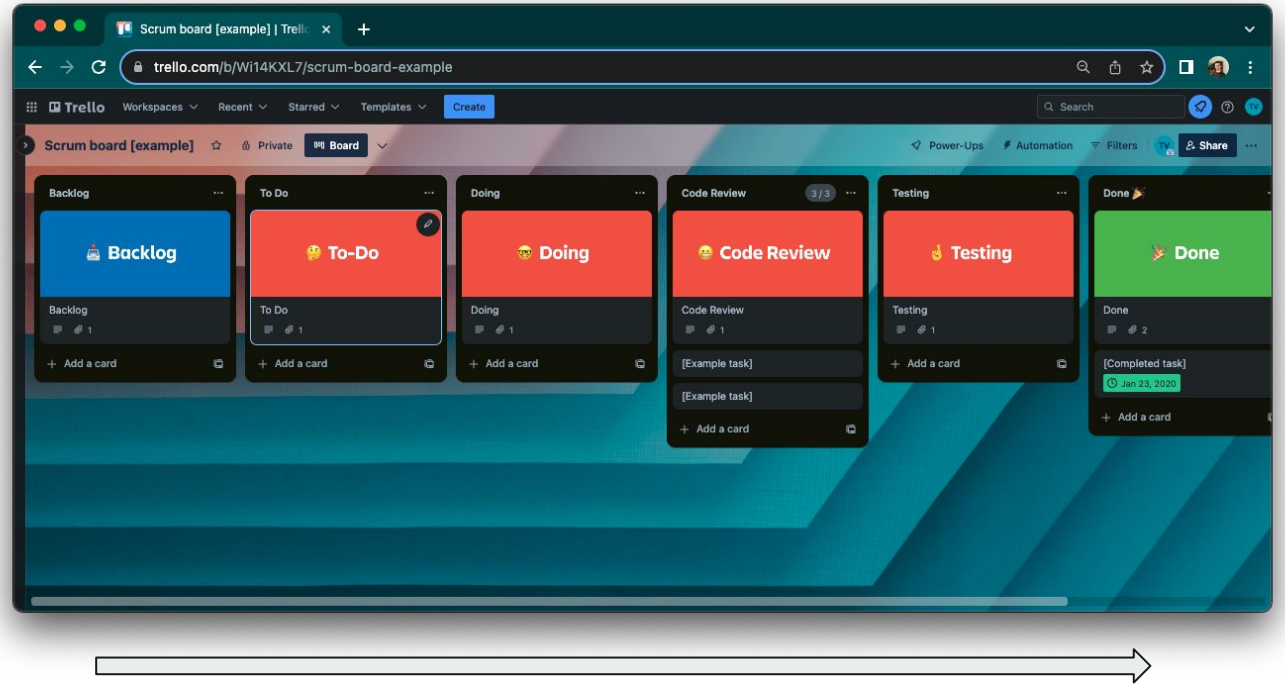
https://www.atlassian.com/agile/kanban

Kanban board

The kanban board is filled with kanban cards (aka tickets or tasks).

New tasks are added to the backlog.

They are then gradually moved along the board.



https://trello.com/

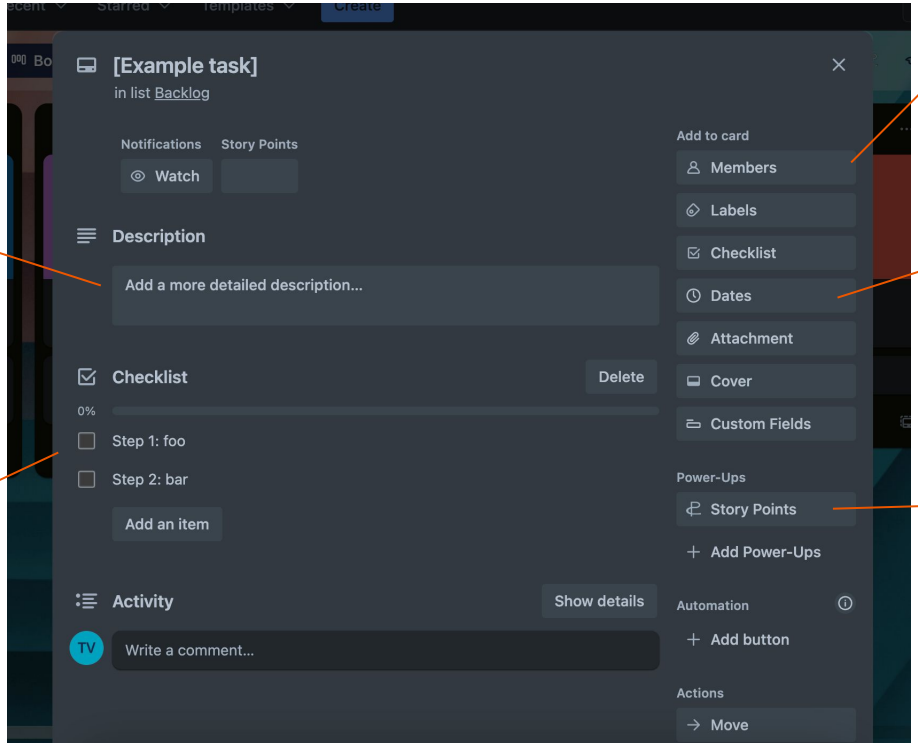
Kanban card

A detailed description of what should be done and how ensures quality of delivery and alignment in the team.

Define task

Include definition of done!

Break up tasks into steps



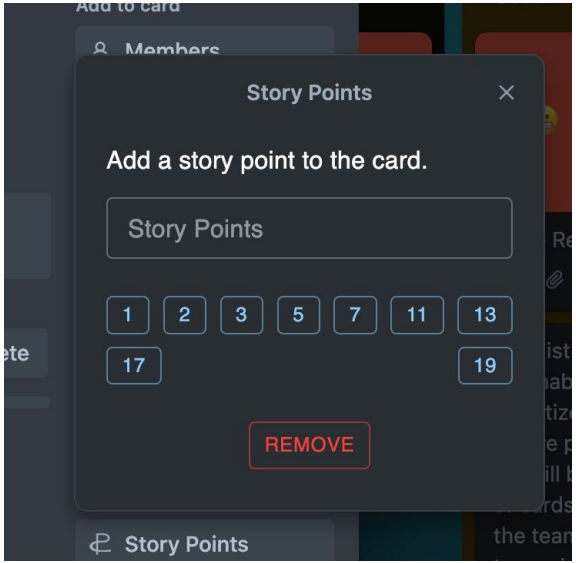
Assign task

Ideally timebox (not always)

Define efforts...

LIÈGE université

Kanban card



Story points are a measure of effort and complexity of a task.

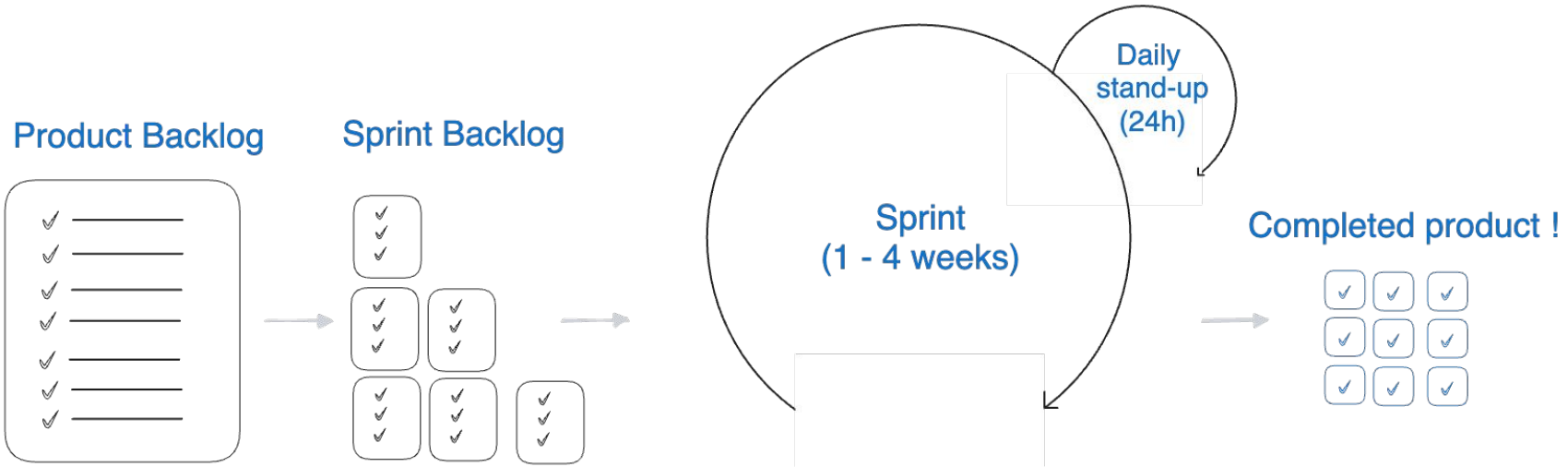
Follows the fibonacci scale.

Sometimes voted on (e.g. https://www.pointingpoker.com/)

If you want more information…

https://www.youtube.com/watch?v=CD0y-aU1sXo

Scrum methodology



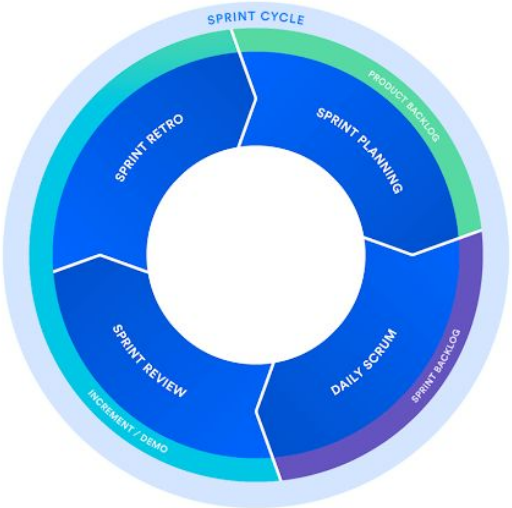
LIÈGE université

Scrum methodology

Sprint organisation

Scrum is organised around **1-4 weeks sprint cycles** (often *2 weeks*).

- **Sprint planning**: Before each sprint, plan and add tasks from the product backlog to the sprint backlog.
- **Daily stand-up**: (Aka daily scrum) During the sprint, have dailies to update the scrum board, similar to kanban board. Go over each team member's progress.
- **Sprint review**: Casually present the work that was done in the last sprint to the rest of the team (definition, celebration and transparency).
- **Sprint retrospective**: Feedback on what went well and what can be improved regarding last sprint.



https://www.atlassian.com/agile/scrum

Why are “daily stand-up” meetings called that way?

Why are “daily stand-up” meetings called that way?

Goal is to make the meeting efficient.

All team members have to answer a fixed set of questions such as:

- What did you do yesterday?
- What did you do today?
- What, if anything, is blocking your progress?

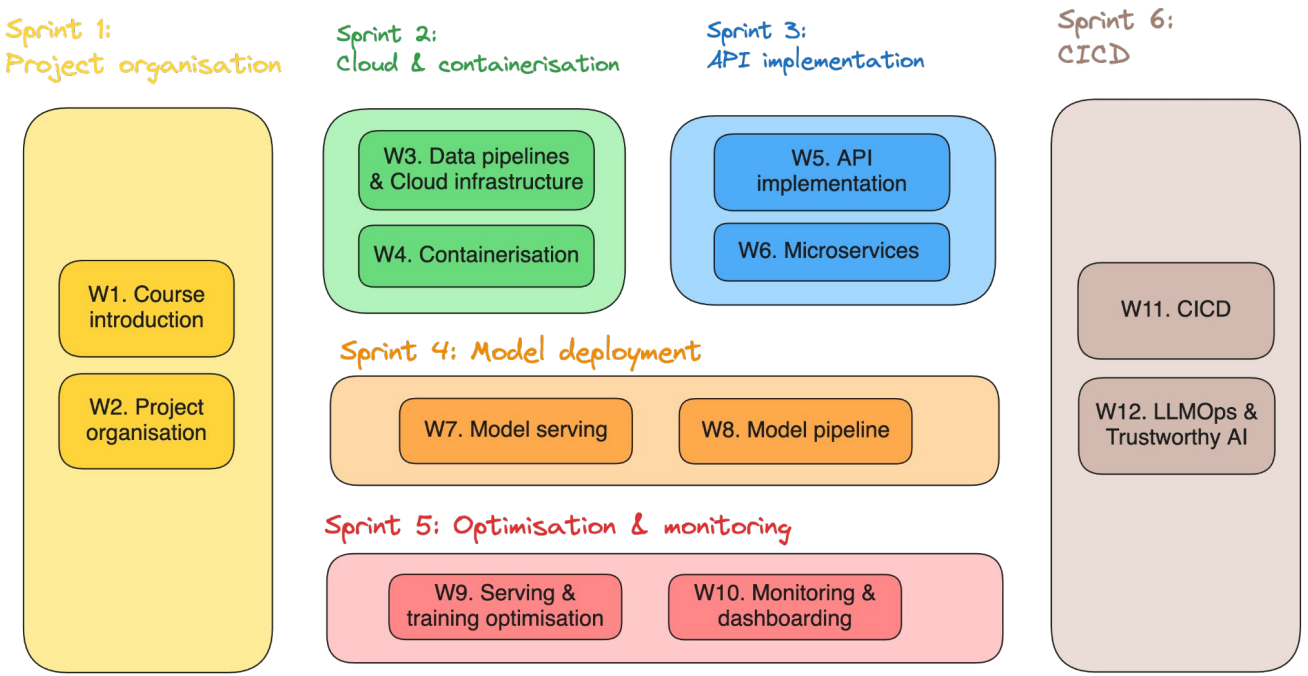
The scrum team

Roles & responsibilities

- **Product Owner**: Represents stakeholders' interests. Selects what is relevant to work on considering product objectives. Define user stories, priorities backlog and decide on product's direction.
- **Scrum Master**: Coach for the team. Ensures the best following of the scrum framework. Maintains the scrum board and backlog, facilitates and leads meetings and addresses obstacles.
- **Scrum Development Team**: Cross-functional developer team (data scientists, ML Engineers, data engineers, DevOps, front-end engineers, …). Delivers a quality product.

https://www.atlassian.com/agile/project-management/scrum-board

… Sounds kinda familiar…



Data sources

Overall data process

Next week

Data source

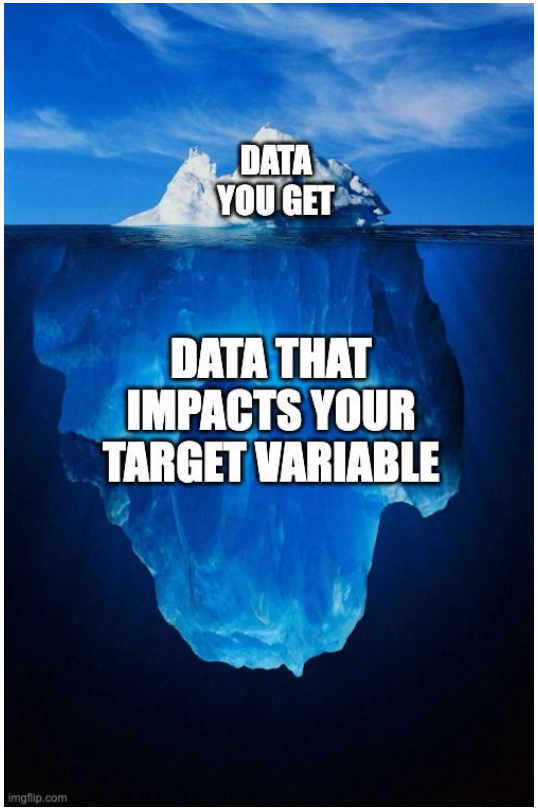
Really important to make sure you understand **all factors** that could be impacting your system.

Spend time discussing with **process experts** and challenge them on what input your model should use.

Example:

- Demand forecasting

  What data would you need to predict demand?



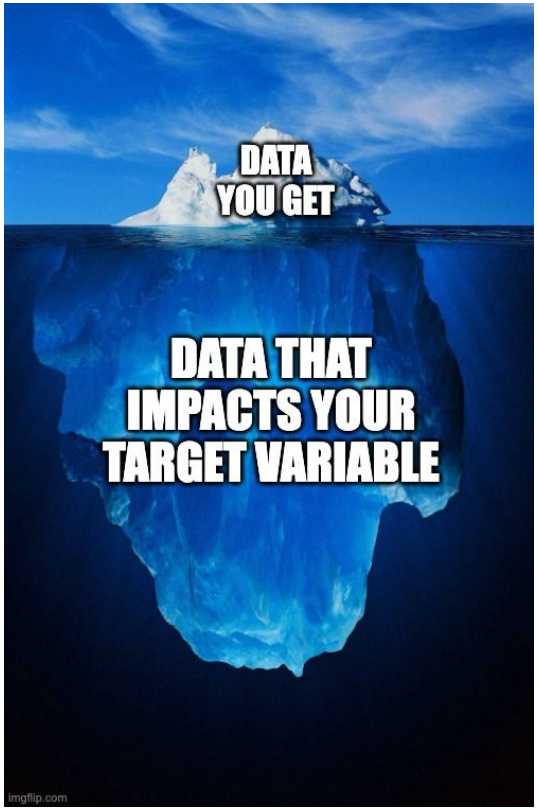
Data source

Really important to make sure you understand **all factors** that could be impacting your system.

Spend time discussing with **process experts** and challenge them on what input your model should use.

Example:

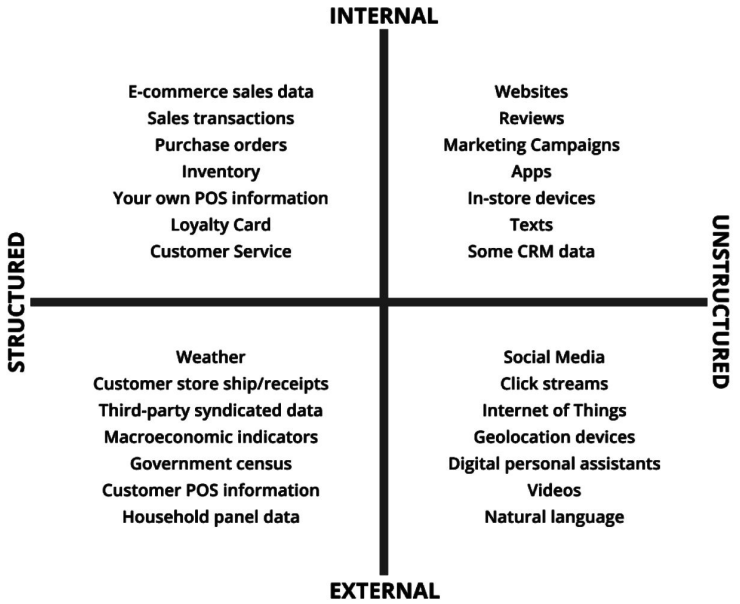
- <u>Demand forecasting</u>
  - It will vary *a lot* depending on industry
    - Fashion → *trends*
    - Ice cream → *weather*
    - Stocks → *macroeconomics*
    - …
  - Ask sales people how they estimate demand.



Data source

Split in two dimensions

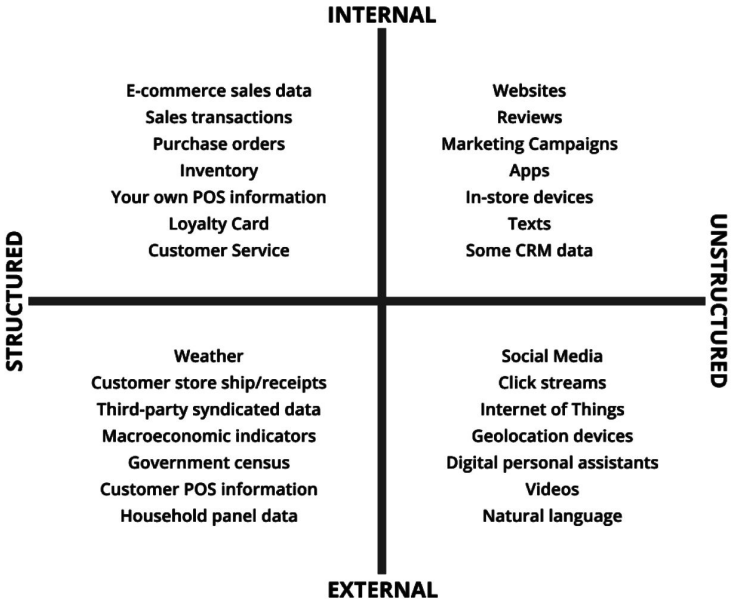
Internal data already belong to the organisation building the model.



External data needs to be gathered, scraped or purchased externally.

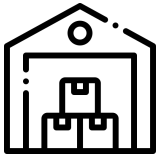
Data source

Split in two dimensions



Structured data

- Usually **tabular organised data**
- **Schema** clearly defined
- Easy to search and analyse
- Schema changes will cause a lot of troubles
- Expensive to setup and store
- Stored in a **data warehouse**



Unstructured data

- Data doesn't have to follow a schema
- All non-tabular data (computer vision, NLP, audio, video, ...)
- Fast arrival
- Cheap to store large volumes
- Can handle data from any source
- Harder to extract information
- Stored in a **data lake**

https://stanford-cs329s.github.io/syllabus.html

Labeled data

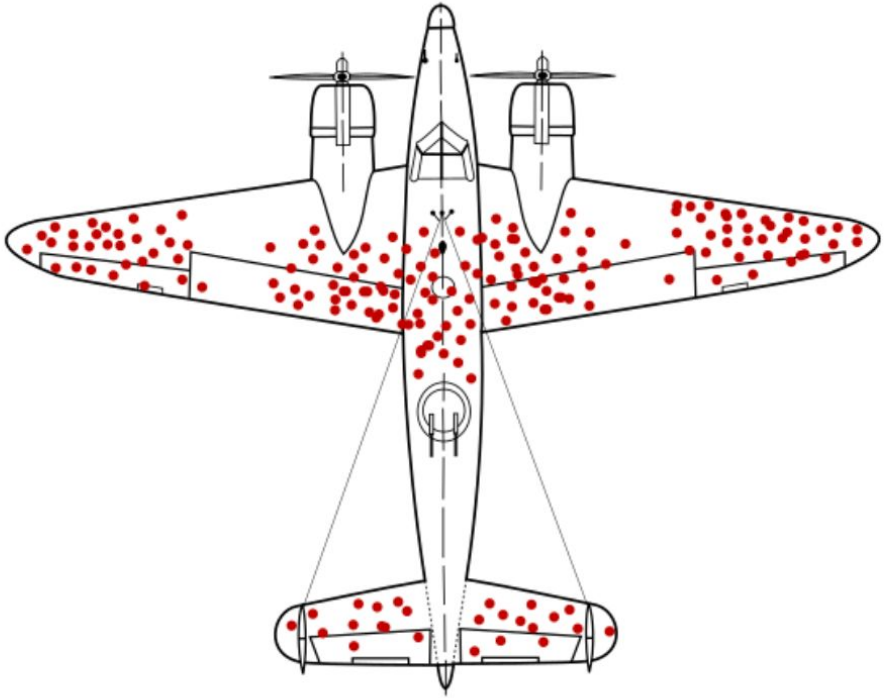
Labeled data is made of observations with a **label** value for what you are trying to predict.

You need

- Enough labeled data
- Good quality labels
- Truly representing what you are trying to model.

Rarely present in the real world…

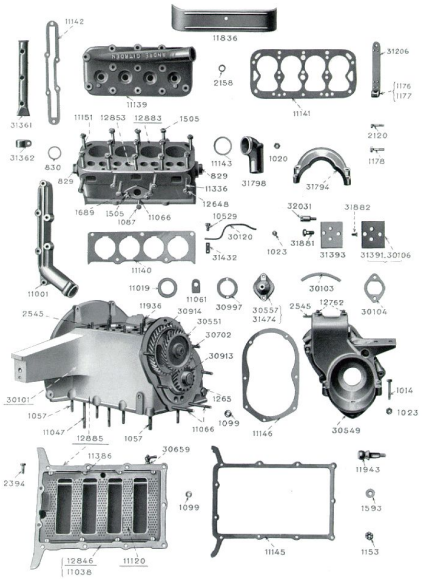
Careful to bias when collecting data



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

Setting up a data collection and/or labelling system

Example 1: Building a physical system



Context: Company selling hundreds of different *engine parts*.

Goal: Automatically classify physical parts using computer vision.

How would you approach it?

Setting up a data collection and/or labelling system

Example 1: Building a physical system

Context: Company selling hundreds of different *engine parts*.

Goal: Automatically classify physical parts using computer vision.

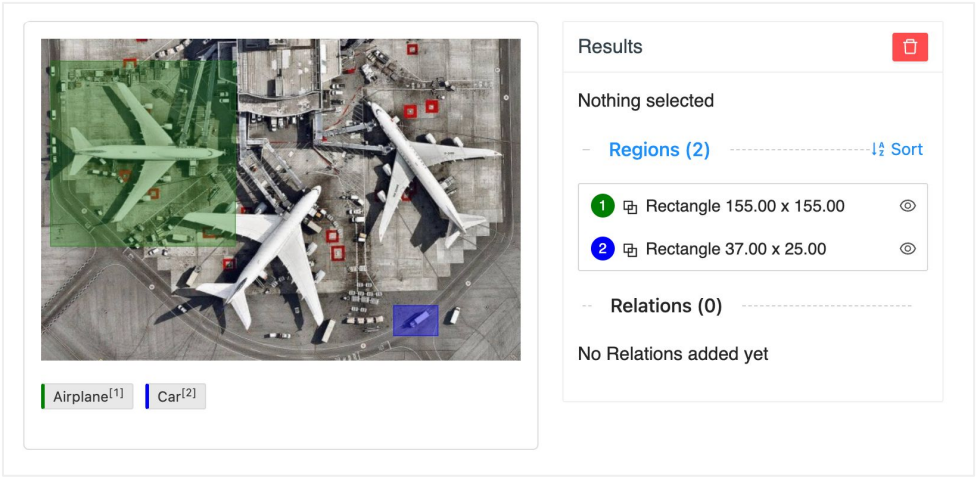
Approach:

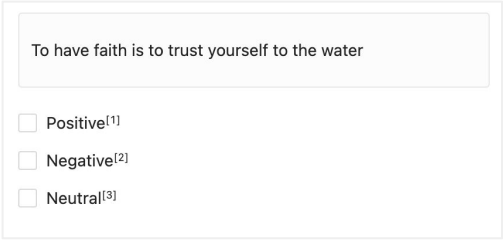
1. Build a box with clear lighting and multiple cameras
2. Build a UI to easily take a snap picture and enter product code
3. Each new and returning products had to go through it for a summer

Result:
>100k product images.

Setting up a data collection and/or labelling system

Example 2: Build a labeling tool and hire a 3rd party company (or student workers)





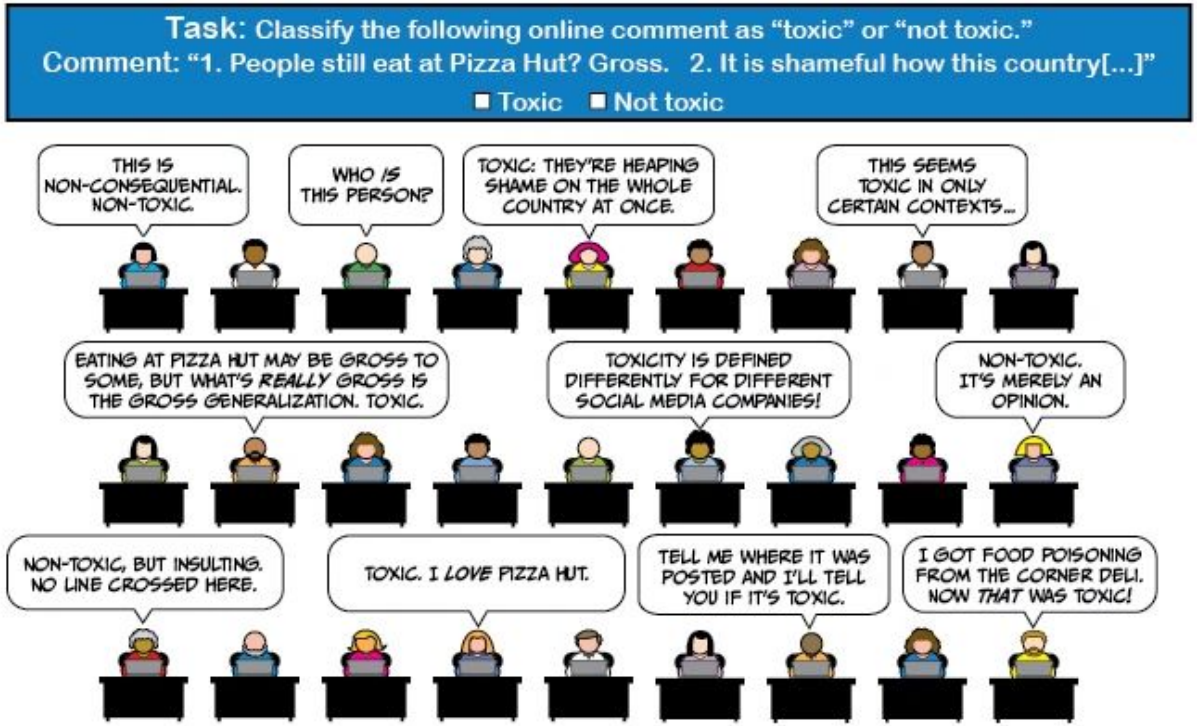
Label Studio

https://labelstud.io/playground/

LIÈGE université

Quality challenge in manual data labelling

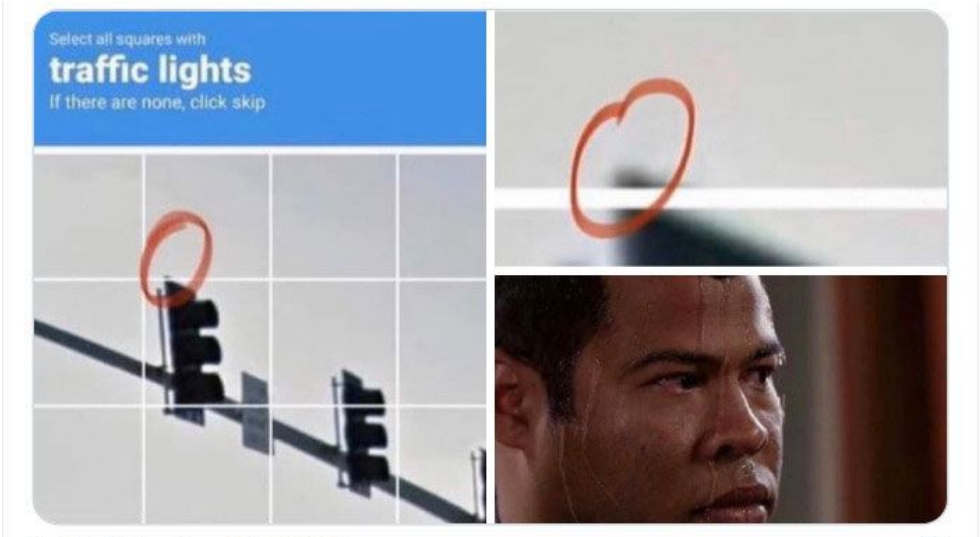
Beware of bias and data quality when collecting the data! Surprisingly hard to get a lot of people to label in a consistent way - inherent bias…



LIÈGE université

Quality challenge in manual data labelling

Beware of bias and data quality when collecting the data! Surprisingly hard to get a lot of people to label in a consistent way - inherent bias…



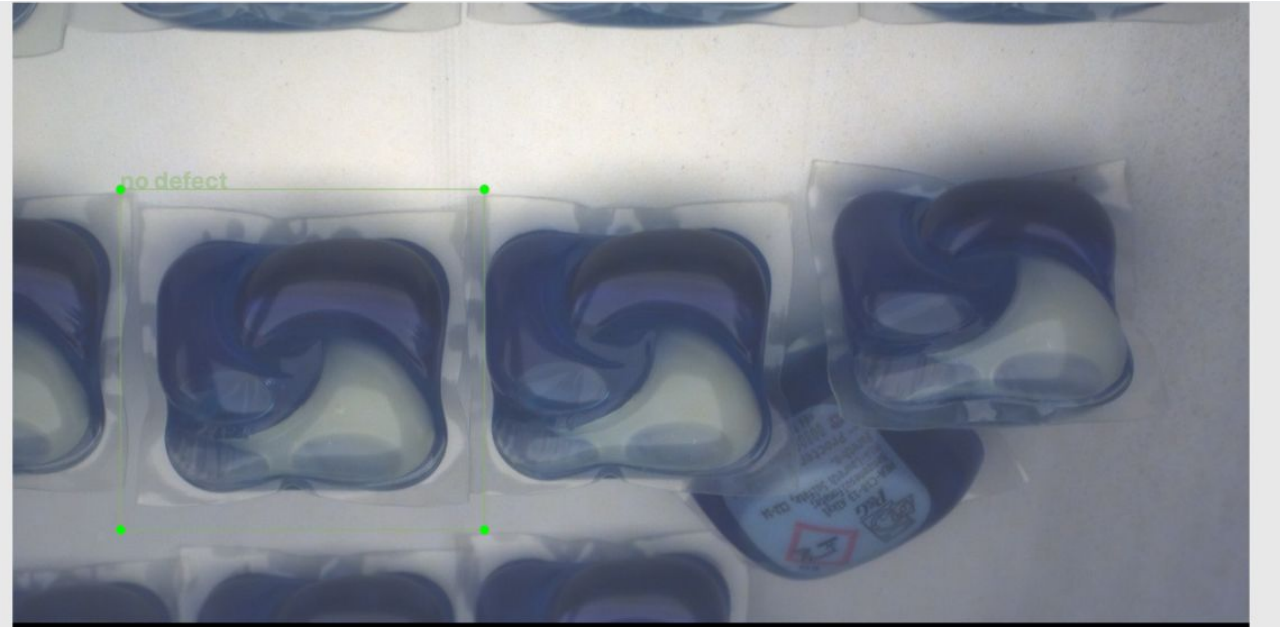
Quality challenge in manual data labelling

Beware of bias and data quality when collecting the data! Surprisingly hard to get a lot of people to label in a consistent way - inherent bias…

Solutions:

- Set strict **guidelines**
  - Explain in detail what should happen
  - Give a lot of examples
- **Review sessions** of difficult examples with different labelers
- **Overlap samples** which are then labeled by multiple labelers
  - Remove or average observations with different labels
- **Calculate bias**
  - Standard deviation of labels
  - Distribution

Be really careful with your labeling strategy as it’s hard to come back from it…



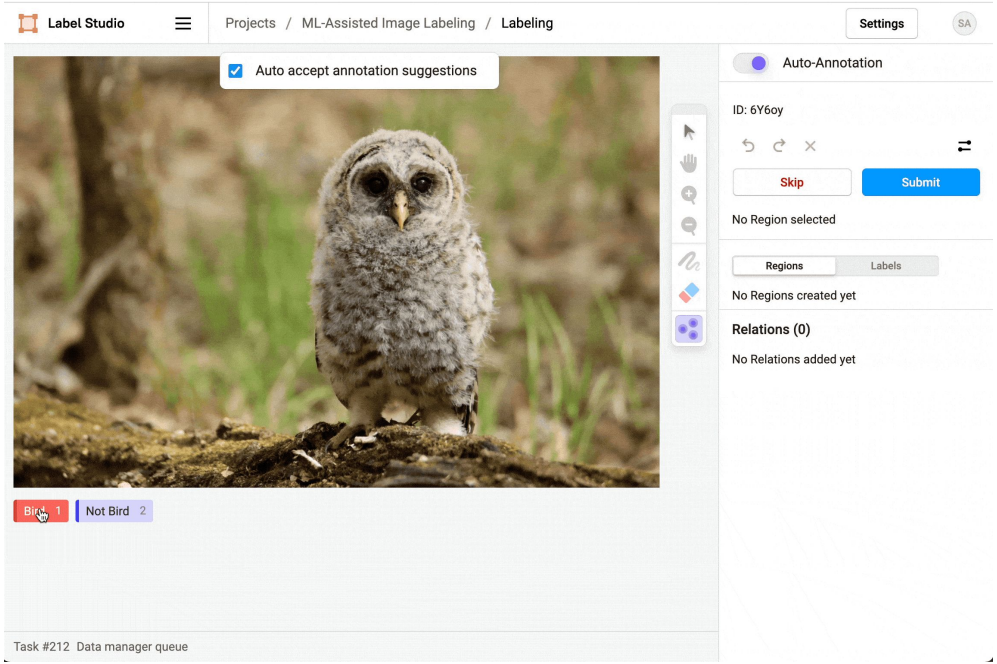
Detecting defect in dishwasher tab pouches.

Did not label overlapping pouches…

Data pseudo-labelling

Use an unsupervised general model to produce some possible labels and get them verify.

Can lead to overfitting on general model !

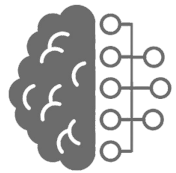


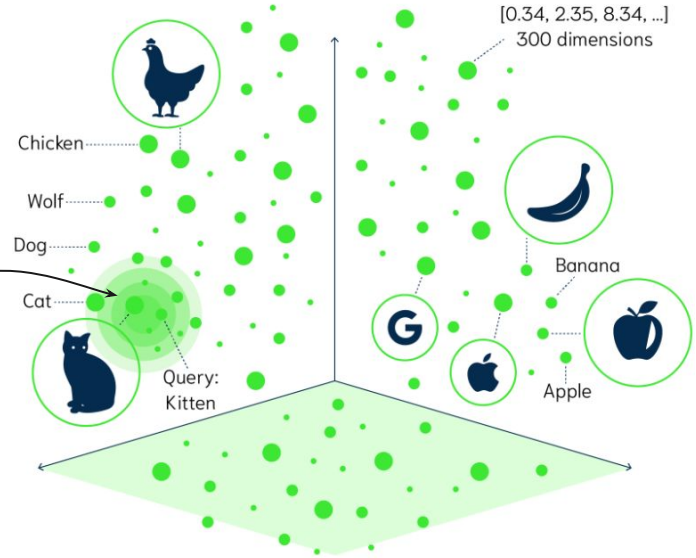
https://labelstud.io/guide/ml.html

Breakout exercise

Semantic Search works with **text embedding**. An ML model is trained to represent any text in a **vector space** where texts tackling similar topics will be close to each others.

Query:
"How often should I feed my cat?"





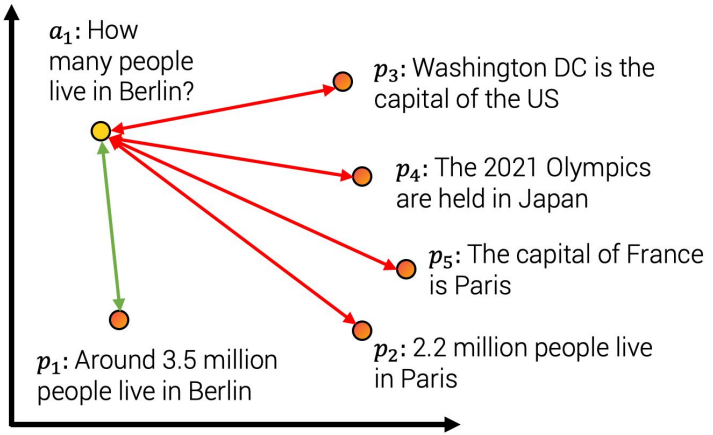
https://labelstud.io/guide/ml.html

Breakout exercise

You can find **pre-trained** embedding models (OpenAI, Huggingface, Cohere, …).

You can **fine-tune** these models on your own data to make them better at understanding **domain specific language**.

What you need: A data set of **Question-Answer pairs**.



Breakout exercise

Exercise:

- Team of 3-4 students
- You got 10min

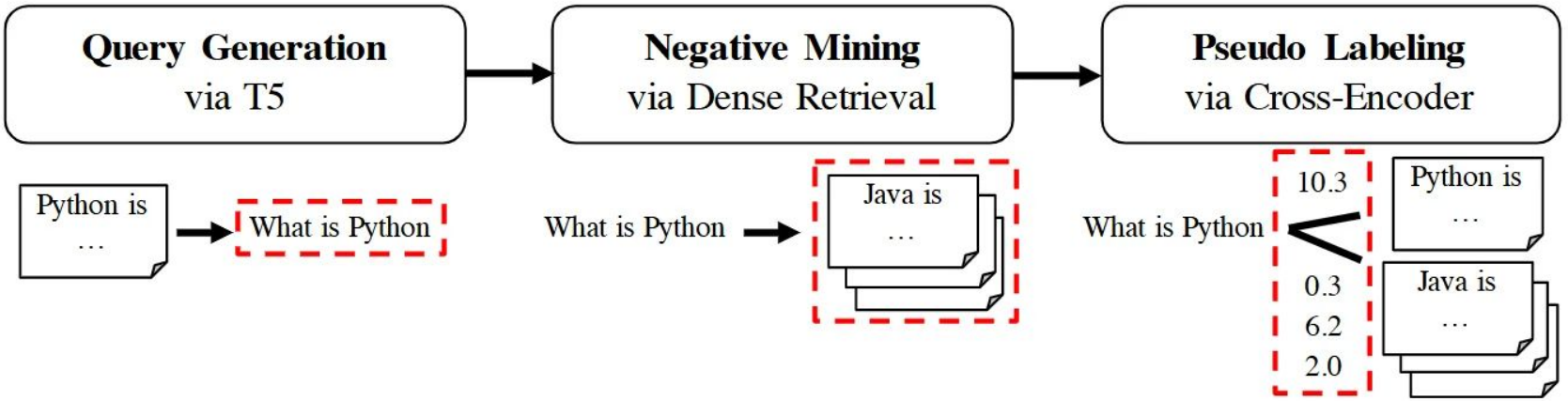
Context:

- Assume you work for a **publishing company** that has **300k news articles** from the last 30 years
- They want to build a **semantic search engine**
- The documents use domain specific language so you want to **fine-tune a text embedding model**
- For that, you need a labeled data set of **Question and Answer pairs**
    - We want it to specifically use the **language** and **domain** of the newspaper publisher
    - The publisher only has documents

Goal:

- Try to come up with a couple of strategies to **create a labeled dataset**

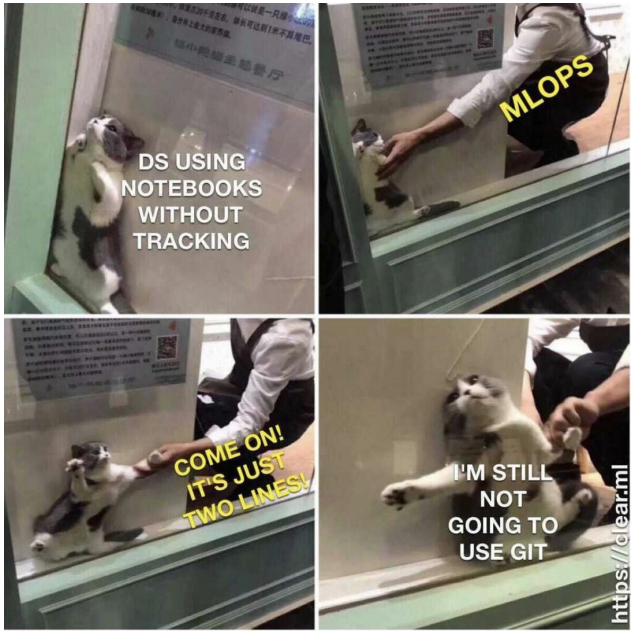
Generative Pseudo Labeling



Code versioning & conventions

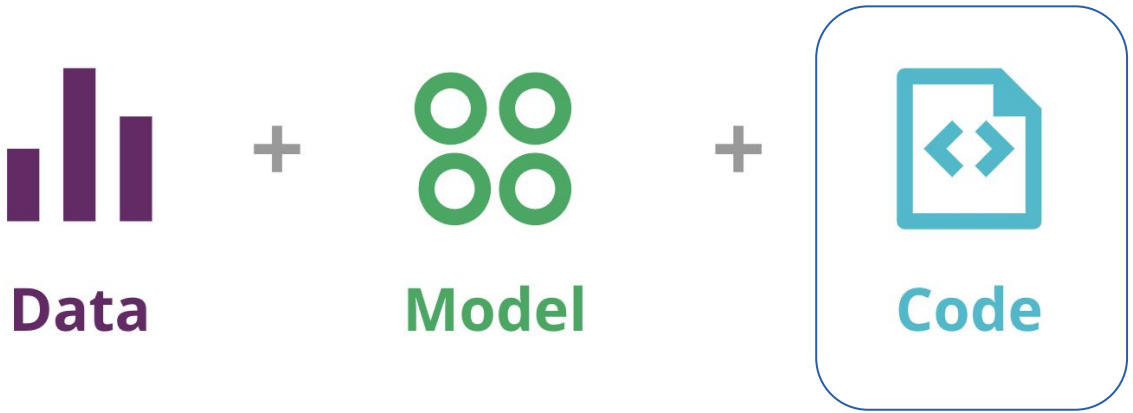
Why does versioning matter?

Name	Date modified	Type	Size
final-draft	8/1/2017 3:48 PM	Microsoft Word D...	16 KB
final-draft2	8/3/2017 3:49 PM	Microsoft Word D...	12 KB
final-draft-final	8/3/2017 10:22 PM	Microsoft Word D...	13 KB
final-modified	8/8/2017 3:49 PM	Microsoft Word D...	13 KB
real-final-draft	8/9/2017 2:22 PM	Microsoft Word D...	13 KB
real-final-draft-v2	8/10/2017 12:33 PM	Microsoft Word D...	16 KB
final	8/11/2017 11:59 PM	Microsoft Word D...	72 KB



LIÈGE université

Versioning applied on different parts of your solution.



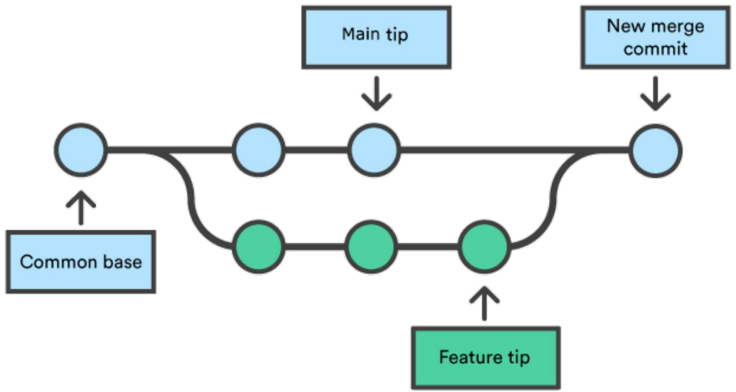
Focus for today!

Code versioning

- Tracking, organising and controlling changes in the code. Enables:
  - Collaboration
  - Track history
  - Revert
  - Versioning
- **Git** is the standard solution for code versioning
- Different Source Code Management (SCM) tools exist such as:
  - Github
  - Gitlab
  - Bitbucket
  - …

Git branching and merging

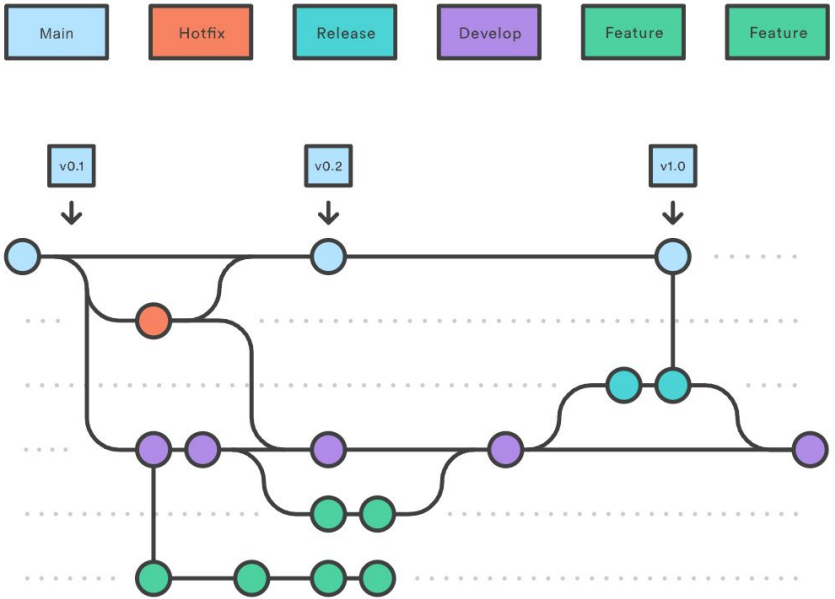
- **Goal**: work at the same time with multiple versions of a repository
- A branch is an independent line of development
  - As default, a repository has a unique branch called **main**
  - Another branch is used to work at changes and extensions before a merge with the main
- For each issue X (feature, bug, etc.):
  - **Create** a new branch X
  - **Implement** X
  - **Merge** X changes with the main



GitFlow

4 types of primary branches

1. **Main** branch stores the official release history (commits with a **version number**)
2. **Develop** branch serves as working branch for new features
3. **Feature** branch gets created each time a specific new features needs to be implemented
4. **Release** cycle - only bug fixes and documentation added to this branch
5. **Hotfix** to quickly patch and maintain production release



https://nvie.com/posts/a-successful-git-branching-model/

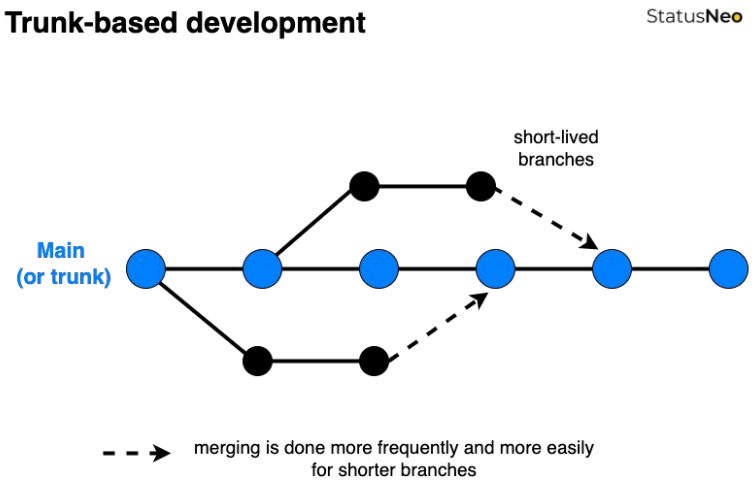
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Product versioning convention

Git Trunk-based development

- Developers merge **small**, **frequent** updates to a core “**trunk**” or main branch
- Based on automatic building and testing of the solution
- Requires CICD
- In trunk-based development, code review should be performed immediately (within a day)
- Somewhat aligned with Agile mindset

Some advocates name it as the new thing…



https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development

GitFlow vs Trunk based

GitFlow	Trunk based
Large feature branches	Small, frequent updates to a core “trunk”
Active collaboration	CICD
Multiple base branch	Automatic testing
Large PRs	Small and frequent PRs
Different phases of development (dev, release, hotfix, …)	No planned code freeze or pause of integration
Still very popular	Documented as the next best thing

Often, teams operate in between.

Traditional Gitflow framework while trying to work with as small branches as possible.

LIÈGE université

GitFlow vs Trunk based

GitFlow	
Large feature branche	core “trunk”
Active collaboration	
Multiple base branch	
Large PRs	
Different phases of de hotfix, …)	pause of integration
Still very popular	st thing

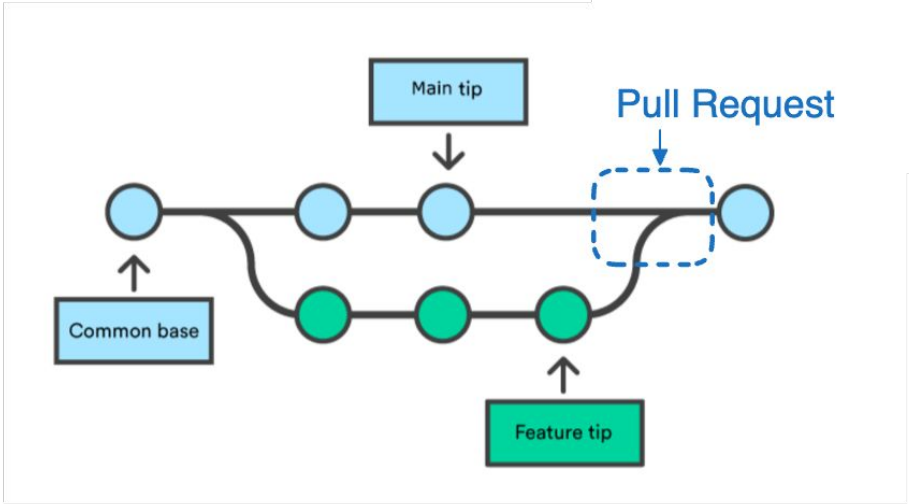
Often, teams operate in between.

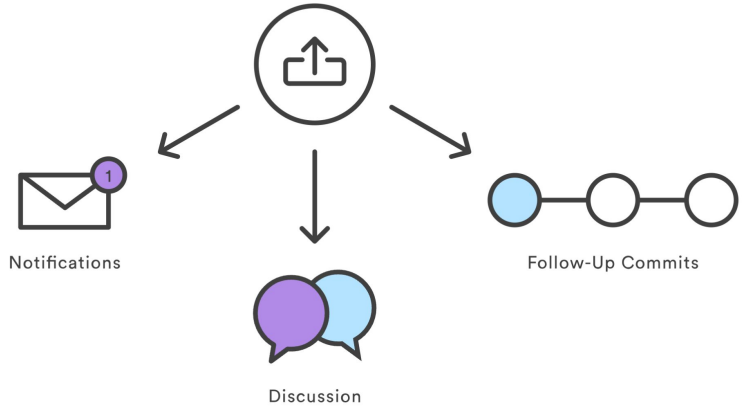
Traditional Gitflow framework while trying to work with as small branches as possible.

LIÈGE université

Pull requests

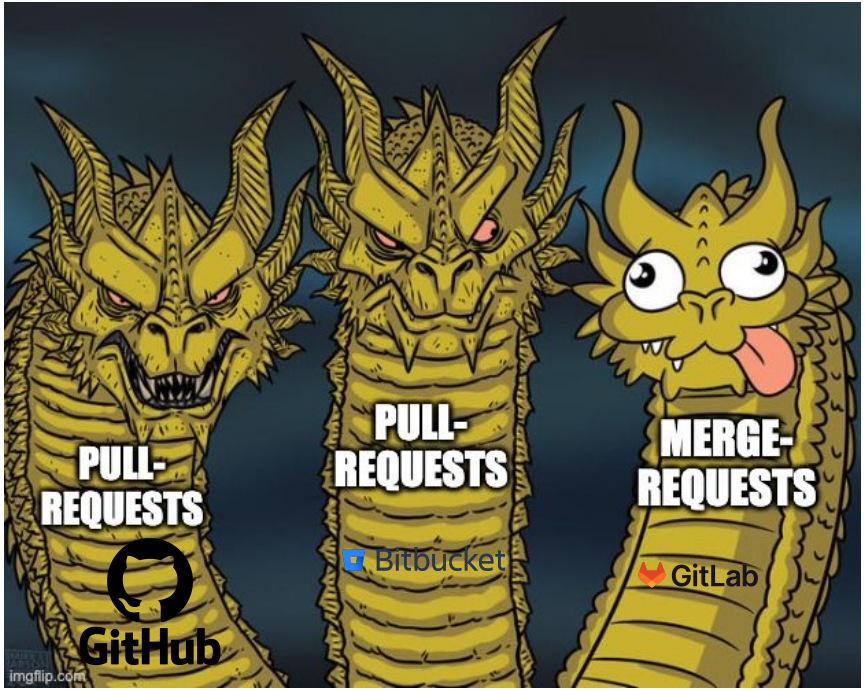
- **Pull Request** (aka **Merge Request**) ⇒ Process to merge branch into main
- Select members that need to **review the codes** before the branch is merged to main.





What is the correct name…

Pull request == Merge request → Synonyms



Pull requests: Code review

Why it is important

- Benefits of code review:
  - **Quality Assurance**: Is this actually the best implementation?
  - **Knowledge sharing**: Aligned team on how the whole code base works.
  - **Component alignment**: Possible catch of logic flaws outside of the feature being reviewed.
  - **Coding standards**: Consistent over code base
  - **Mentorship**: Guidance for less experienced developers
  - **Documentation**: Guarantees that codes are readable
  - **Team morale**: We work as a group, less isolated features
- Tech-lead needs to **dedicate time to code review**

Strict correlation between developer team maturity and enforcement of code reviews!

Be smart - shouldn't inder team efficiency (especially in early stages of a project)

https://google.github.io/eng-practices/review/

Pull requests: Code review

What reviewers should look at

- **Design**: Is the code well-designed and appropriate for your system?
- **Functionality**: Does the code behave as the author likely intended? Is the way the code behaves good for its users?
- **Complexity**: Could the code be made simpler? Would another developer be able to easily understand and use this code when they come across it in the future?
- **Tests**: Does the code have correct and well-designed automated tests?
- **Naming**: Did the developer choose clear names for variables, classes, methods, etc.?
- **Comments**: Are the comments clear and useful?
- **Style**: Does the code follow our style guides?
- **Documentation**: Did the developer also update relevant documentation?

https://google.github.io/eng-practices/review/

Importance of documentation

Method/script level documentation (docstring)

Used to document your function. *“Would I be comfortable that someone else can easily reuse this code?*

Describe the general **functionality**, the input **arguments** and what it **returns**. Example styles:

```
"""
This is a reST style.

:param param1: this is a first param
:param param2: this is a second param
:returns: this is a description of what is returned
:raises keyError: raises an exception
"""
```

reStructuredText (reST)

```
"""
This is an example of Google style.

Args:
    param1: This is the first param.
    param2: This is a second param.

Returns:
    This is a description of what is returned.

Raises:
    KeyError: Raises an exception.
"""
```

Google

Importance d

Method/script level

Used to document yo

euse this code?

Describe the general

tyles:

```
"""
This is a reST style.

:param param1: this is a firs
:param param2: this is a seco
:returns: this is a descripti
:raises keyError: raises an e
"""
```

reStructuredText (

What is wrong with the docstring in this code snippet?

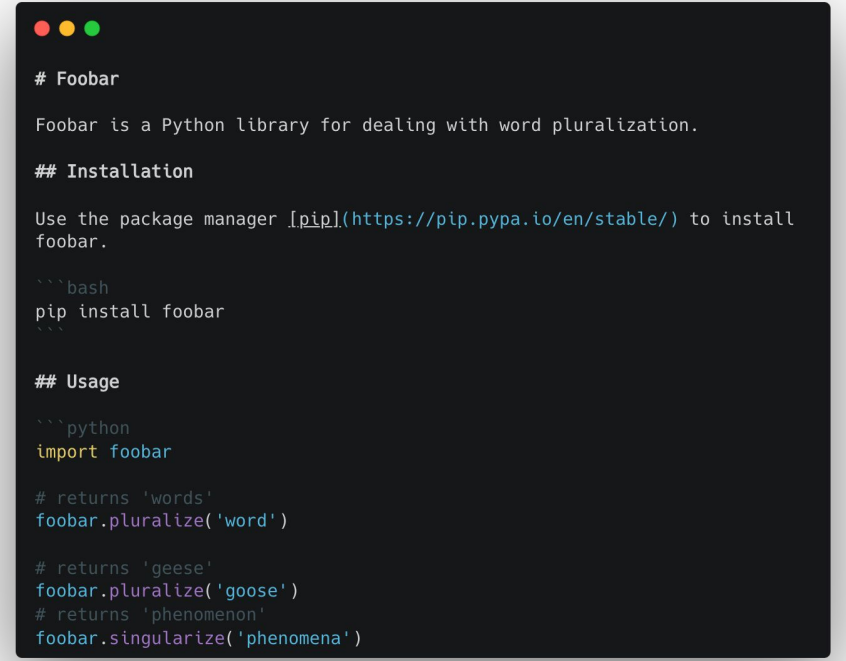

```
def calculate_area(length, width):
  '''Calculates the area of a rectangle. This method takes two arguments, the length and
width of the rectangle, and returns the area.
  Arguments:
  length - the length of the rectangle.
  width - the width of the rectangle.
  Returns: The area of the rectangle calculated using the formula length*width.'''
  return length * width
```

Importance of documentation

Component/system level (markdown)

Used as general documentation (e.g. README.md)

Important to clearly document the **structure** of your repository and how to **use it**.



````
# Foobar

Foobar is a Python library for dealing with word pluralization.

## Installation

Use the package manager [pip](https://pip.pypa.io/en/stable/) to install
foobar.

```bash
pip install foobar
```

## Usage

```python
import foobar

# returns 'words'
foobar.pluralize('word')

# returns 'geese'
foobar.pluralize('goose')
# returns 'phenomenon'
foobar.singularize('phenomena')
````
````

Importance of documentation

Component/system level (markdown)

Used as general documentation (e.g. README.md)

Important to clearly document the **structure** of your repository and how to **use it**.

Rubber ducking: Tell yourself what exactly you're doing here. Good way to voice a problem. (Usually for debugging but also for documentation).

```
# Foobar

Foobar is a Python library for dealing with word pluralization.

## Installation

Use the package manager [pip](https://pip.pypa.io/en/stable/) to install
foobar.

```bash
pip install foobar
```

Usage

```python
import foobar

# returns 'words'
foobar.pluralize('word')

# returns 'geese'
foobar.pluralize('goose')
# returns 'phenomenon'
foobar.singularize('phenomena')
```
```

Importance of documentation

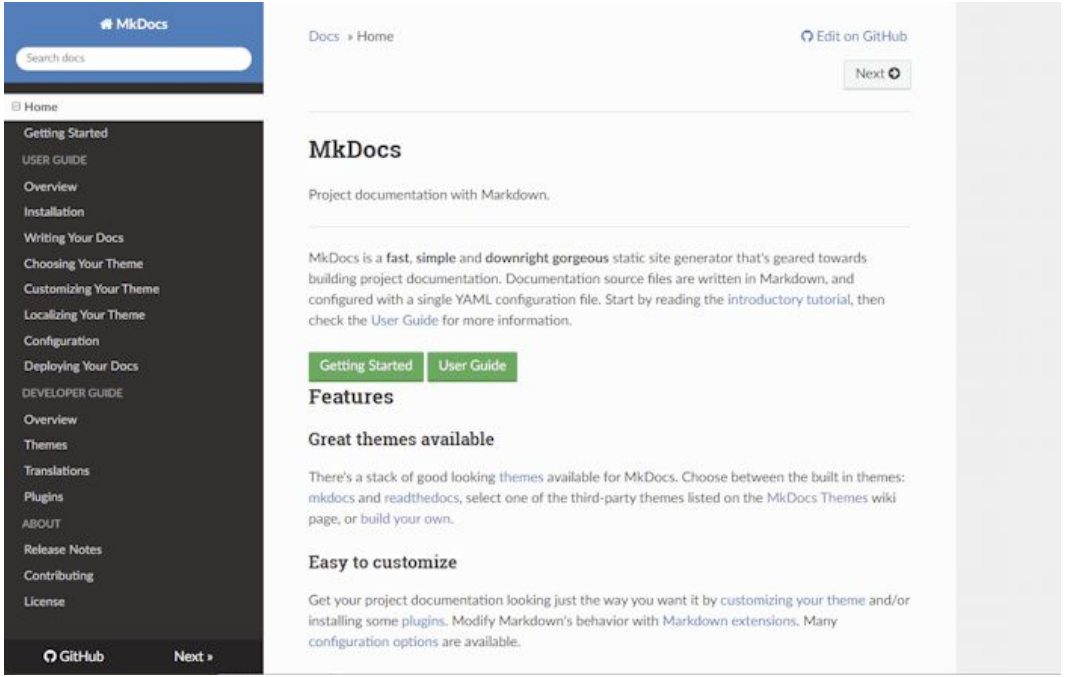
Component/system level (markdown)

```
mkdocs.yml
docs/
    index.md
    about.md
    license.md
```

You can use tools such as **MkDocs** or **docusaurus** to produce a full documentation page from your directory.

It takes in a certain structure of documentation and will generate an HTML page.

Identifies docstrings and other types of documentation directly from your codebase.



https://www.mkdocs.org/

The .gitignore file

Issue with Git

- Pain to erase memory
  - *Don’t ever push any sensitive information on git!*
- Limited storage space (~2 GB)

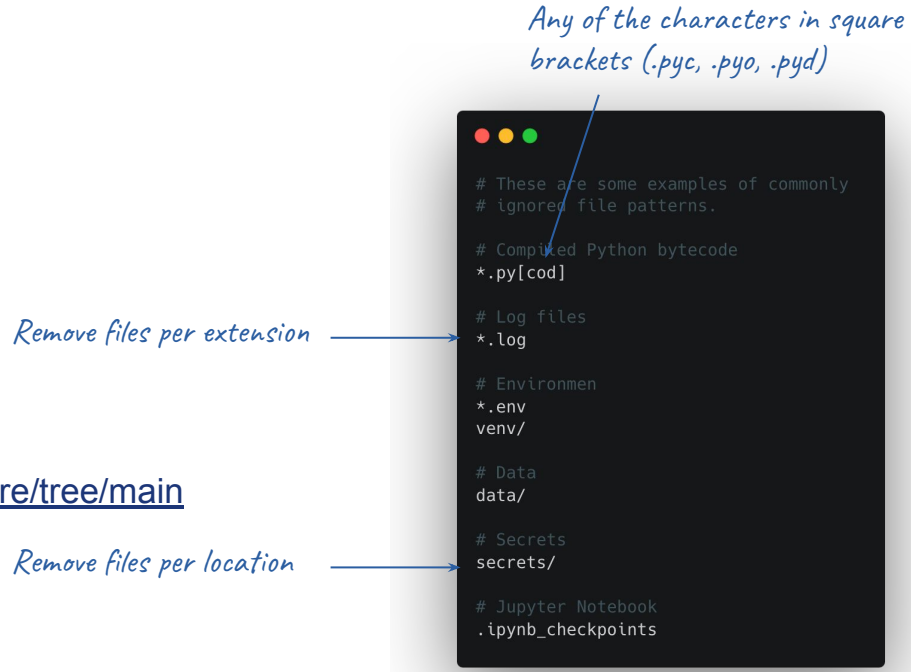
The .gitignore file

Issue with Git

- Pain to erase memory
  - *Don't ever push any sensitive information on git!*
- Limited storage space (~2 GB)

.gitignore file!

- Patterns for ignoring files to be submitted to git
- Usually set with standard file patterns
- Many examples: https://github.com/github/gitignore/tree/main



Scripting vs notebooks

Attributes of scripts:

- **Stateless**
  - Scripts do *not* save variables and what you run in the **global state**. Therefore they are **stateless** (unlike notebooks which are **stateful**)
  - Scripts require passing variables to functions and classes. Healthy to **structure** your implementation.
- **Linear**
  - Scripts run everything **from beginning to end linearly**, which takes away the risk of running cells in the wrong order
- **Reproducibility**
  - Easier to **reproduce** and **run tests** with different parameters with scripts.

"Only a sith deals in absolutes"

⇒ notebooks aren't bad, they just serve a different purpose ⇒ **Experimentation** (which requires being messy)

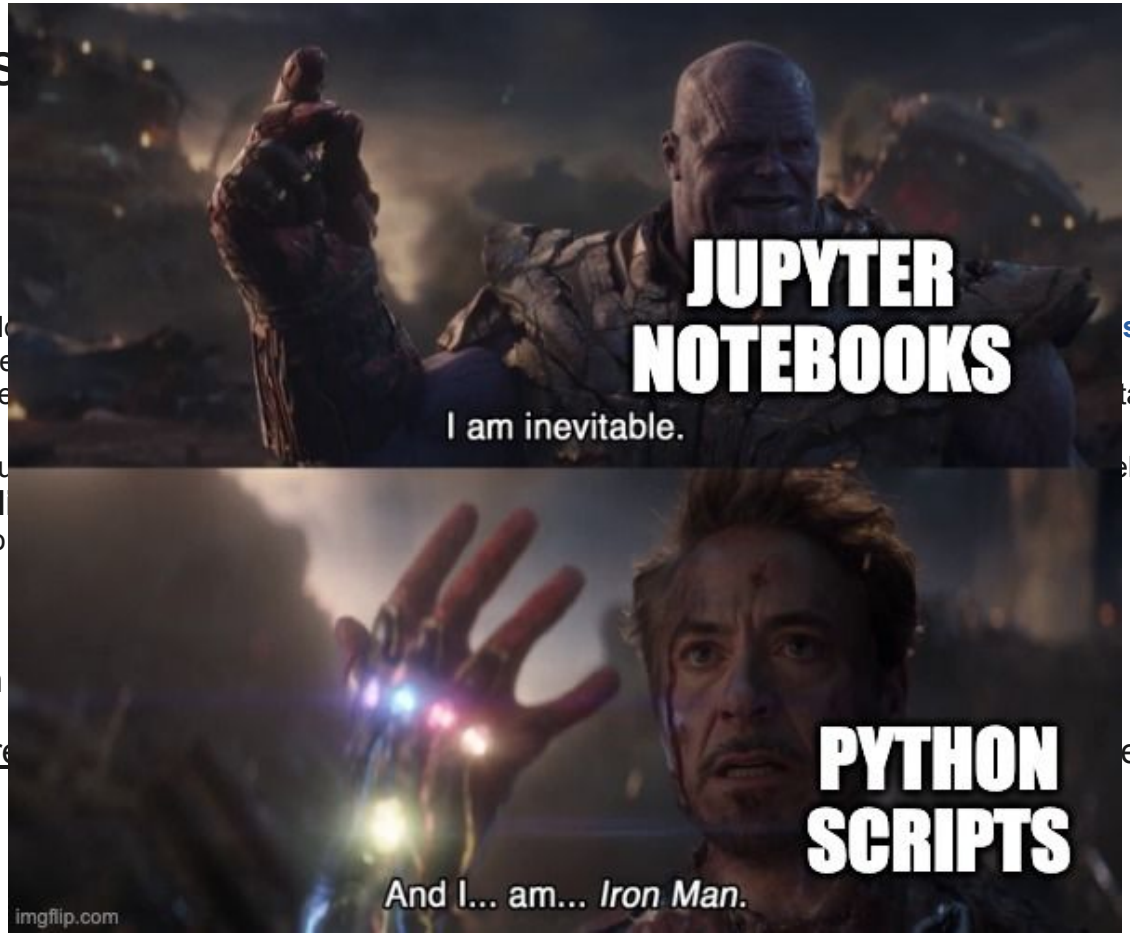
Scripting vs

Attributes of scripts:

- **Stateless**
  - Scripts d… **s** (unlike notebooks which are…
  - Scripts re… …tation.
- **Linear**
  - Scripts ru… …lls in the wrong order
- **Reproducibil…**
  - Easier to…

"Only a sith deals in …

⇒ notebooks are… …equires being messy)



LIÈGE université

https://madewithml.com/courses/mlops/scripting/

Code convention: Python PEP8

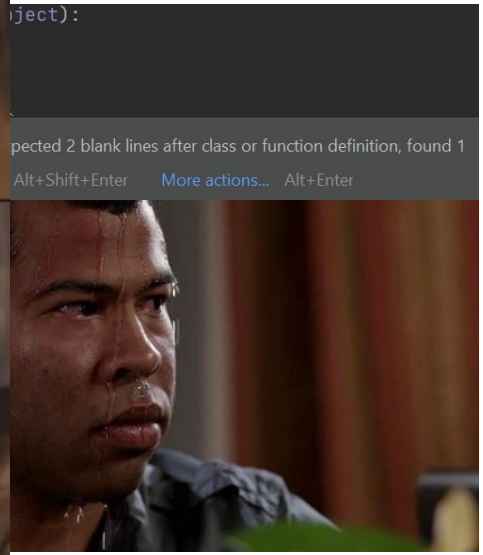
- Ensures a consistent code quality across a team.
- Set of rules on styling, indenting, naming conventions and documentation
- Integrate/automate in your code editor (IDE) to make it easy. Often enforced PEP8 during pull request submission.
- 🕵️ Developers can be judgy… Make your life easy, adapt clean codes

Tools to automatically check compliance with PEP8 will be covered in the CICD lecture

https://peps.python.org/pep-0008/

Code convention: Python PEP8

- Ensures a consis
- Set of rules on st
  documentation
- Integrate/automa
  Often enforced P
- 🕵️ Developers ca
  codes



https://peps.python.org/pep-0008/

Code convention: Python PEP8

- Code Layout and Indentation
  - Use 4 spaces per indentation level. Avoid mixing tabs and spaces
- Line Length
  - Limit lines to 79 characters, and 72 characters for comments or docstrings
- Blank Lines
  - Surround top-level function and class definitions with two blank lines. Use a single blank line to separate logical sections within a function or method.
- Imports
  - Group imports into three sections: standard library, third-party libraries, and local application/library-specific imports. Each group should be separated by a blank line, and avoid wildcard imports (from module import *).
- Naming Conventions
  - Use snake_case for variable and function names, UPPERCASE for constants, and CamelCase for class names. Avoid single-character variable names, except for counters or iterators.
- Comments and Docstrings
  - Write clear and concise comments that explain why something is done, not how. Use triple double-quotes (""") for module, class, and method docstrings.
- Function and Variable Annotations
  - Use type hints for function arguments and return values (e.g., def add(x: int, y: int) -> int:).
- Exceptions
  - Use specific exception types rather than a generic except: block. Always clean up resources (e.g., files) with context managers like with open().

https://peps.python.org/pep-0008/

What is wrong with this code snippet?


```python
import sys, os

def some_function(x,y):
    print('x+y=',x+y)
    if (x==0):
        print("x is 0!")
        x=x+1
    for i in range(0,10): print(i)
    try:os.listdir('.')
    except Exception as e:print(e);sys.exit(1)
```

What is wrong with this code snippet?


```python
import sys, os

def some_function(x,y):
    print('x+y=',x+y)
    if (x==0):
        print("x is 0!")
        x=x+1
    for i in range(0,10): print(i)
    try:os.listdir('.')
    except Exception as e:print(e);sys.exit(1)
```

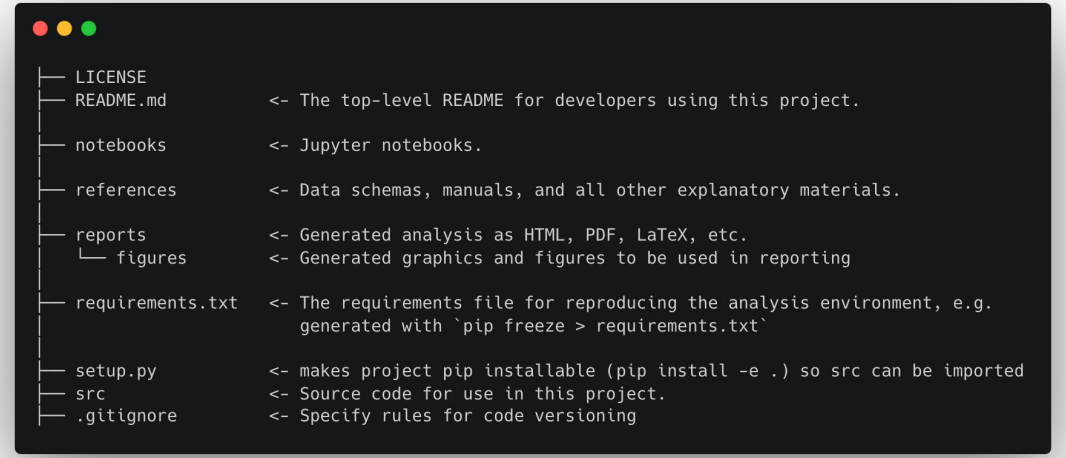
https://docs.bigchaindb.com/projects/contributing/en/latest/cross-project-policies/python-style-guide.html

Cookie cutters: Initiating your repository

Many **cookie cutters** exist to easily set up specific kinds of code repositories.

The Cookiecutter Data Science project sets up a repository structure tailored to data science projects.

⚠️ **Disclaimer**: A bit old and subjective. E.g. saving data locally is not best practice.



https://drivendata.github.io/cookiecutter-data-science/

Wrap-up

Lecture summary

Topic	Concepts	To know for…	
		Project	Exam
Agile / Scrum	• Methodology • Types of roles & meetings • Kanban board		Yes
Data sources	• Internal vs external • Structured vs unstructured • Data labelling techniques		Yes
Code versioning	• Git Flow • Trunk based • Pull requests, feedback giving • PEP8	Yes	
Demo: Git code versioning		Yes	

Project objective for sprint 1

Week	Work package	Requirement
W01	Pick a **team** ● Try to mix skills and experience ● If you didn't find one let one of the teachers know and we'll allocate you to one	Required
W02	Select a **use case** ● Previous course ● Kaggle Datasets ● … Make sure to pick a use case where **data is available**. Ideally pick something with interesting data and a real world application.	Required
W02	**Define** your use case. Fill in a ML Canvas template page (You can skip the *Inference* part as we will tackle that in a later sprint.)	Required
W02	Find a **cool name** for your project ✨	Required
W02	Setup **communication channel** (Discord, trello)	Required
W02	Setup a **code** versioning **repository** ● We recommend Github as we will cover Github Actions during this course	Required
W02	Submit your project by sending a filled in project card to the teaching staff with basic information about your project. We might give you some feedback and ask for parts to be changed.	Required

Project deliverables

Deliverables

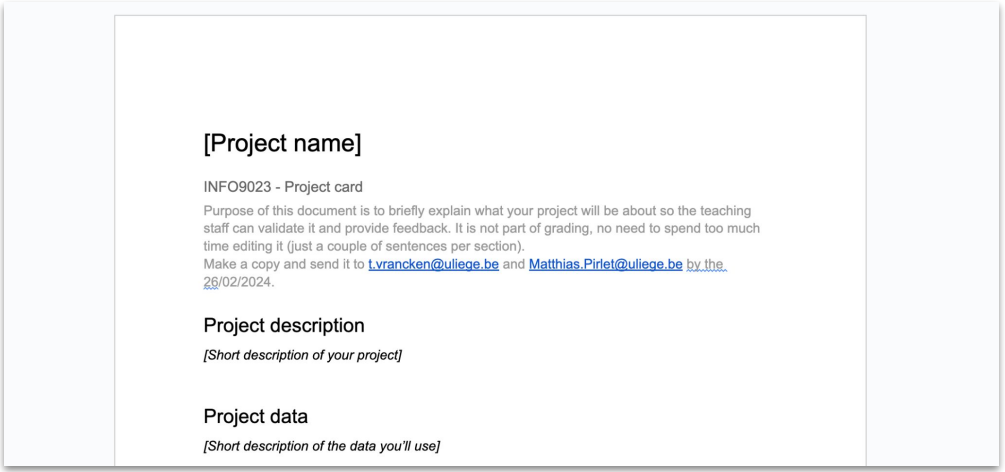
- **3 Milestone (MS) meetings**:
  - MS 1: Present work from *sprint 1 & 2*
    - Present your general use case (BMC), the data preparation and the result from your model experimentation.
  - MS 2: Present work from *sprint 3 & 4*
    - Present your architecture for model deployment and automated training.
  - MS 3: Overall presentation
    - Whole project, inc. sprint 5
- Each MS presentation will be accompanied with a *code submission on Github*. Teaching staff will not go over all codes but key information can be provided in README + check that everything is well implemented.

No handovers before MS meetings. The work packages per sprints are there to guide you, but you're free to implement it at your pace. Teaching staff is there to provide support every week.

Project deliverables

⚠️ Submit your **team** and **project** by filling in this project card template (make a copy) and sending it to the teaching staff.

- Purpose is for the teaching staff to validate the project ideas and potentially provide feedback.
- Fill it in before next lesson.



[Project name]

INFO9023 - Project card

Purpose of this document is to briefly explain what your project will be about so the teaching staff can validate it and provide feedback. It is not part of grading, no need to spend too much time editing it (just a couple of sentences per section).
Make a copy and send it to t.vrancken@uliege.be and Matthias.Pirlet@uliege.be by the 26/02/2024.

Project description

[Short description of your project]

Project data

[Short description of the data you'll use]

LIÈGE université

Lab: Git code versioning

LIÈGE université