Project
Implementation of a (Big) Data Management Backbone

Overview and Framework

Big Data Management – FIB – UPC

Data Science Projects

Data science projects require creating systems that deploy data pipelines spanning three different areas:

- Business understanding (domain)
  - What do we want to analyse?
  - What is the added value of an analytical question for the organisation?
- Data management
  - Data discovery
  - Data modeling
  - Data storage
  - Data processing
  - Data querying
- Data analysis
  - Data preparation
  - Modeling
  - Validation
  - Explainability
  - Visualization

Example of Bad Practices

Data lifecycle at a real Polish company (as it was):

- The company collects measurements on thermal energy consumption of their customers and use it to predict energy consumption
  1. Data is generated by several independent devices. Data is stored locally and periodically sent as a text file to the company server. Data is then integrated with weather data at the server-side (join by date)
  2. From all this data, they generate a monthly dataset containing measurements (per day) and weather data. For years 2016-2020 it meant 160 CSV files.
  3. Over these files, they conducted outlier detection and standarization scripts (i.e., data preparation) written in Python and executed at the server side.
  4. The final CSV files are stored in a local directory, which are then read by Python scripts that prepare the data and train a model using *scikit-learn* (https://scikit-learn.org/stable/).
  5. Validation and its interpretation is conducted using some data not used for the training, using 10-cross fold validation. The model is then visualized to facilitate its interpretation using *matplotlib* (https://matplotlib.org/).
  6. Data scientists iterated throughout this pipe several times until obtaining a satisfactory model persisted as a *scikit-learn* model using *pickel* (https://docs.python.org/3/library/pickle.html).

This pipeline is usually implemented via notebooks

Exercise 1: Looking for clusters visually

From the course *Transition to Data Science*. Buy the entire course for just $10 for many more exercises and helpful video lectures.

You are given an array `points` of size 300x2, where each row gives the (x, y) co-ordinates of a point on a map. Make a scatter plot of these points, and use the scatter plot to guess how many clusters there are.

Step 1: Load the dataset *(written for you)*.

In [1]:
```
import pandas as pd

df = pd.read_csv('../datasets/ch1ex1.csv')
points = df.values
```

Step 2: Import PyPlot

In [2]:
```
import matplotlib.pyplot as plt
```

Step 3: Create an array called `xs` that contains the values of `points[:,0]` - that is, column `0` of `points`.

In [3]:
```
xs = points[:,0]
```

Step 3: Create an array called `ys` that contains the values of `points[:,1]` - that is, column `1` of `points`.

In [4]:
```
ys = points[:,1]
```

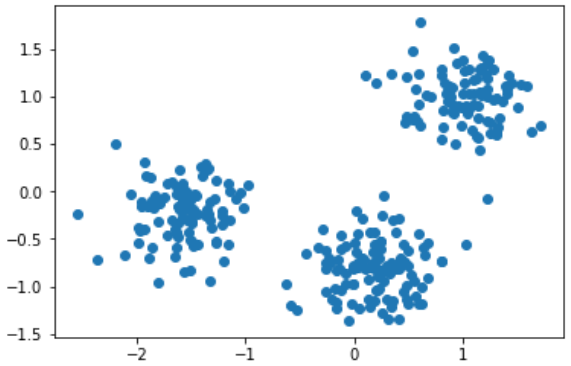
Step 4: Make a scatter plot by passing `xs` and `ys` to the `plt.scatter()` function.

In [5]:
```
plt.scatter(xs, ys)
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x110fc45c0>
```

Step 5: Call the `plt.show()` function to show your plot.

In [6]:
```
plt.show()
```



Read data from temporal files

Use off-the-shelf ML libraries

Pre-process the data to fit the needs of the algorithm input parameters

Run the algorithm

Visualise the result to allow its interpretation

Extracted from: https://github.com/benjaminwilson/python-clustering-exercises/

The Baseline Pipeline

Pros

- Dynamic and integrated environment that includes text, code and visualisations
- Enables ad-hoc (not systematic) sharing of analytical workflows
- Enables trial / error (the data science loop)
- Acces to tons of analytical libraries

Cons

- Most data science libraries process data locally (e.g, in CSV). This approach compromises:
  - Governance
  - Persistence
  - Efficiency in data access
  - Concurrency
  - Reliability
  - Security
  - Performance

NO COMMON BACKBONE

- No code / data sharing / reusage
- No single source of truth (data / code)
- No global optimizations

Operationalizing Data Science Pipelines

Main Challenge

"Quality deliveries, with short cycle time, need a high degree of automation"

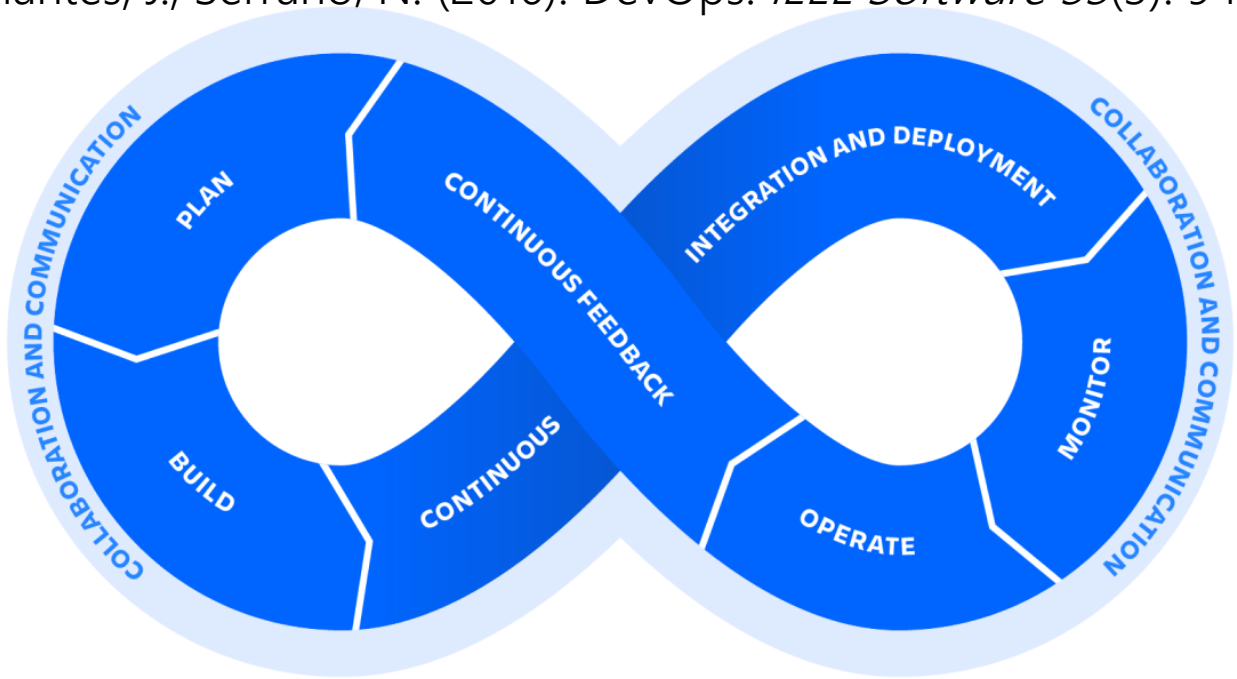
Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. (2016). DevOps. IEEE Software 33(3): 94-100.

DevOps

DevOps is about fast, flexible development and provisioning processes.

It integrates the two worlds of development and operation, using automated development, deployment, and infrastructure monitoring. Its an organizational shift in which, instead of distributed isolated groups performing functions separately, cross-functional teams work on continuous operational feature deliveries.

Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. (2016). DevOps. *IEEE Software 33*(3): 94-100.



The DevOps Lifecycle

- Plan: elicit requirements and prioritize new functionalities to be implemented in iterations. To improve speed and quality, agile methodologies must be applied
- Build: the process transforming standalone code into software artifacts. Build processes include version control, documentation, managing dependencies, code quality, testing and compilation. Build also considers deploying an application to different environments: typically, development (*dev*), testing (*test*) and production (*prod*)
- Continuous integration (CI) and delivery: developers' work must be integrated as often as possible to facilitate testing and reduce time-to-market. CI pipelines orchestrates automated builds, tests and deployment into release workflows
- Monitoring: of the infrastructure and the deployed software in production
- Continuous feedback: monitoring results in a loop back into the next plan iteration and should be considered to plan the ahead iterations

From DevOps to DataOps / MLOps

DevOps ignores a key aspect in Data Science: data and its lifecycle

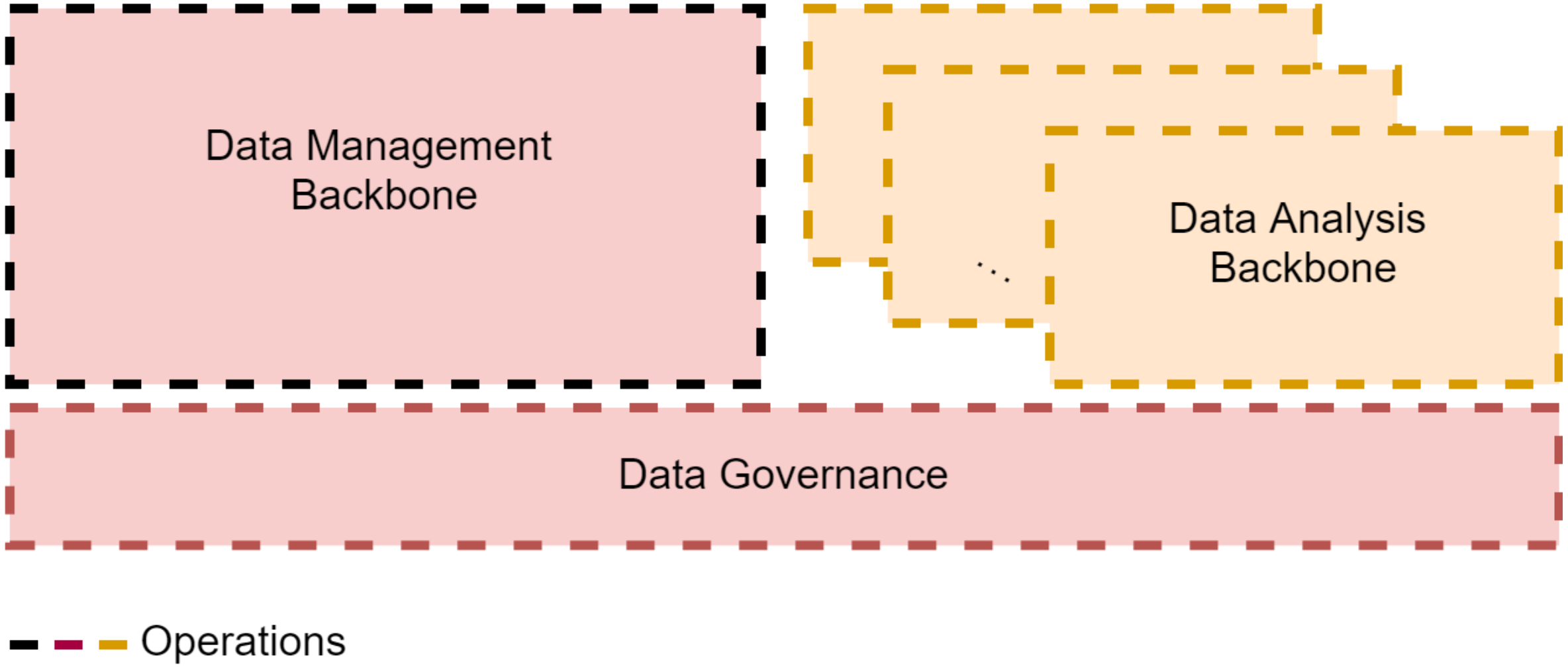
DataOps was introduced to cover this gap by *combining an integrated and process-oriented perspective on data with automation and methods from agile software engineering, like DevOps, to improve quality, speed, and collaboration and promote a culture of continuous improvement.*

Ereth, J. (2018). DataOps - Towards a Definition. LWDA 2018: 104-112.

In short, the DataOps lifecycle is needed to manage the complexity of the data lifecycle in data science projects (data engineering)

***Disclaimer**: you may read about MLOps too. However, in essence, DataOps / MLOps talk about the same problem. Simply, the latter focuses more on the ML part while the former cover the whole data lifecycle. We aim at providing an holistic view in this course and that is why we choose DataOps in front of MLOps*

DataOps in a Nutshell



DataOps in a Nutshell

The data management backbone (common for the whole organisation)

- Ingest and store external data into the system
  - Data Integration (standardization and data crossing)
    - Syntactic homogenization of data
    - Semantic homogenization of data
  - Clean data / eliminate duplicates
- Expose a cleaned and centralized repository
  - Data profiling

The data analysis backbone (repeats for every analytical pipeline)

- Extract a data view from the centralized repository
- Feature engineering
- Specific pre-processing for the given analytical task at hand
  - Labeling
  - Data preparation specific for the algorithm chosen
- Create test and validation datasets
- Learn models (either descriptive stastitical analysis or advanced predictive models)
- Validate and interprete the model

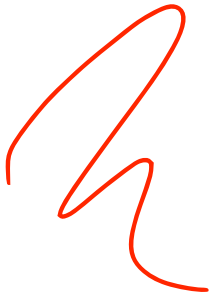
DataOps in a Nutshell

Operations (common per project)

- Data management backbone monitoring: metrics such as processes performance, disk and memory usage, etc.
- Data analysis backbone monitoring: metrics such as model accuracy, training time, fairness, etc.

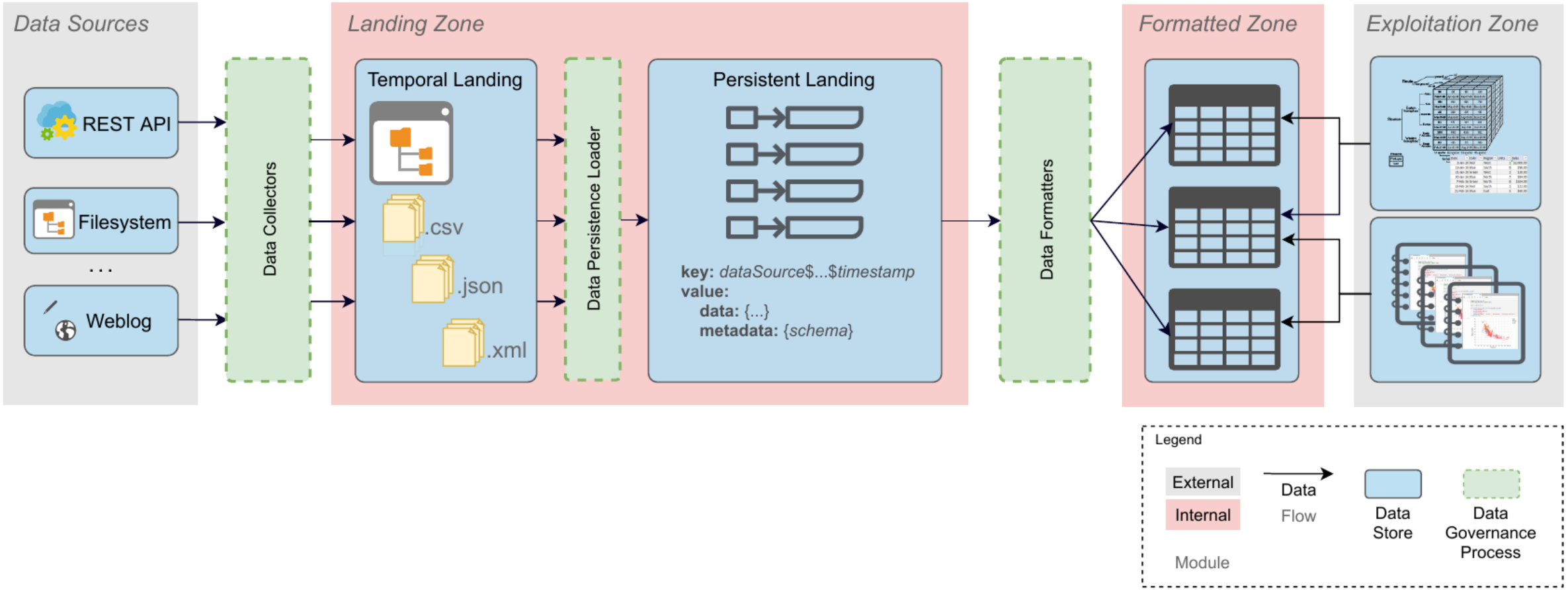
Governance (common for the whole organisation)

- Domain-specific metadata: information about the day-by-day vocabulary used by analysts and their link to data variables / data sources, data preparation tasks for a given analysis, featuresand datasets used by the trained models, etc.
- Domain-independent metadata: information about domain-independent processes conducted on data such data lineage / traceability, etc.
- Physical metadata: information required to automatically process physical records such as data formats / data types, format-specific templates, etc.



The Data Management Backbone

The Data Management Backbone



The Data Management Backbone

- Data is ingested in the landing zone as it is produced (raw data)
  - The temporal landing stores the files temporary, until they are processed
  - The persistent landing tracks ingested files by data source and timestamp (i.e., versions)
- Data is then homogenized, according to a canonical data model in the formatted zone (syntactic homogenization)
  - This is where data cleaning happens. However, only generic data cleaning techniques (i.e., independent of a specific project) occur here
- The exploitation zone exposes data ready to be consumed / analysed either by advanced analytical pipelines or external tools. Two main kinds of tasks are conducted to generate this zone (semantic homogenization):
  - Data integration: new data views are generated by combining the instances from the Formatted Zone. A view may serve one or several data analysis tasks. Relevantly, data integration spans data discovery, entity resolution, ad-hoc transformations and data loading into a target schema in a potentially different data model (not necessarily in the form of the canonical data model)
  - Data quality processes: data quality required to have a wider view (e.g., instances from different sources or requiring to happen after data Integration is conducted)

The Relevance of the Exploitation Zone

It is the zone where data is exposed either to in-house data scientists (to conduct ad-hoc advanced data analysis tasks) or to external tools

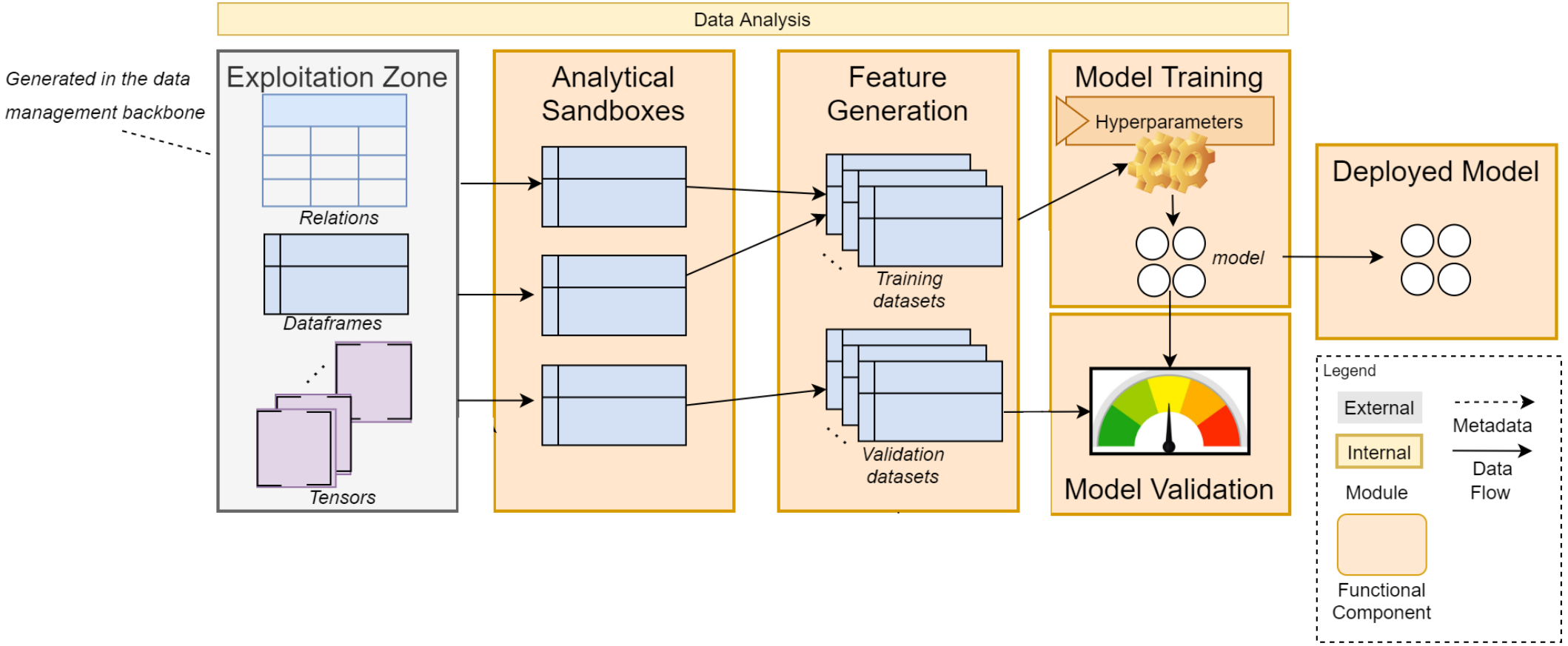
For analytical purposes, the three most extended data models are tensors, relations and dataframes for:

- Query & reporting (descriptive data analysis): require relations (traditional OLAP engines over relational databases) or dataframes (newer engines built on top of Data Lakes)
- Machine Learning and Data Mining (predictive data analysis): typically require dataframes (e.g., R, Data Science Python libraries / frameworks, SAS, etc.). Also, distributed Machine Learning and Data Mining (e.g., MLlib) typically require dataframes. Deep Learning frameworks, however, require tensors to work.

***Disclaimer**: other data models may be required at this stage to connect the exploitation zone to other tools (e.g., graph data modeling). However, we focus on those models required for traditional ML / DM*

The Data Analysis Backbone

The Data Analysis Backbone



The Data Analysis Backbone

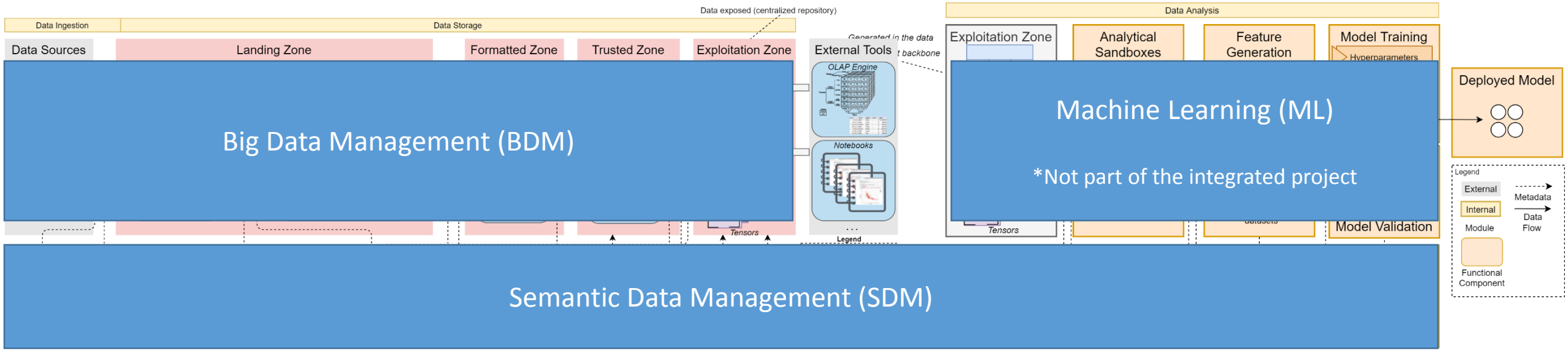
Unlike the data management backbone, there is an analytical pipeline per analysis need. Thus, a project may define several analytical pipelines

- The analytical sandboxes capture subsets of the elements created in the exploitation zone and of relevance for the analysis at hand
- During feature generation, features are generated from the analytical sandboxes. The following tasks take place:
  - Data preparation rules, specific for the algorithm and kind of analysis, are applied
  - Labeling, if required, is also conducted here
  - As result, two corpus of datasets are generated: the training and validation datasets
- Model training requires choosing an algorithm and specifying the required algorithm hyperparameters, and this outputs a model
- Then, the generated model is validated according to some quality criteria (e.g., accuracy, recall, etc.)
- Out of the models generated, one is chosen to be deployed

Scope of the project

Implementation of a (Big) Data Management backbone

Overview

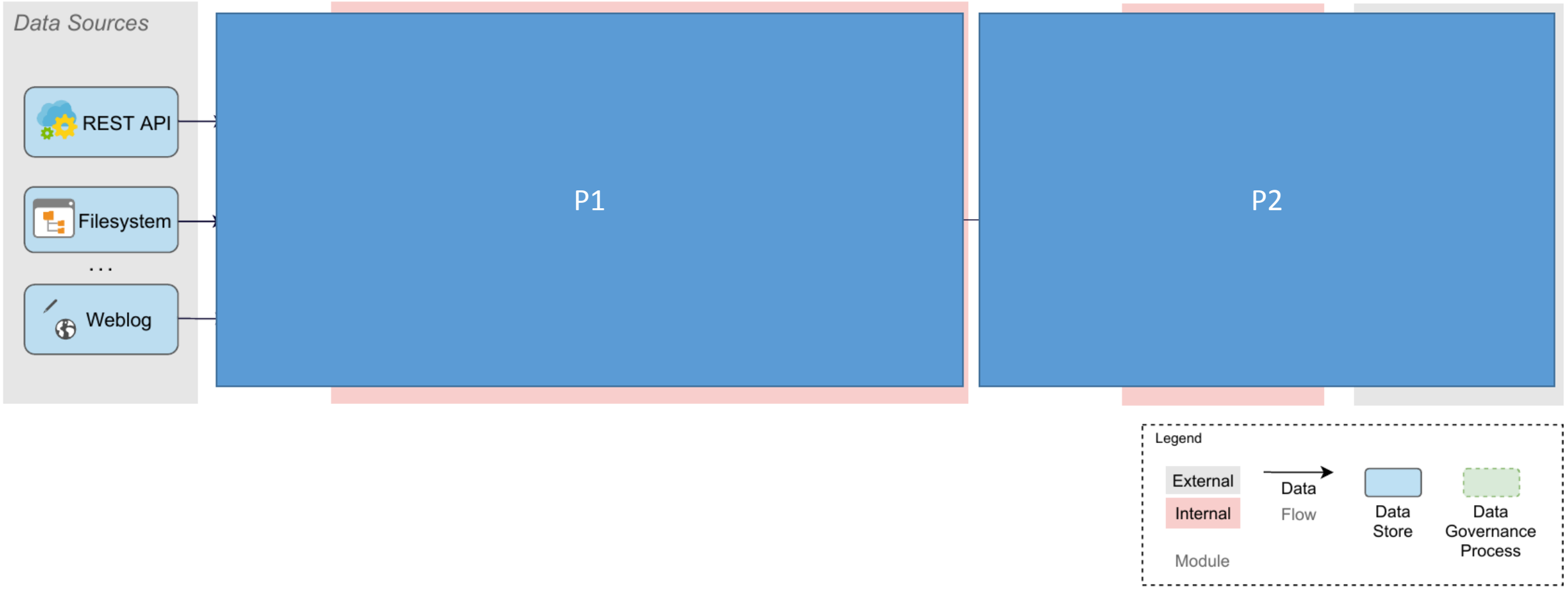


Viability of Business Projects (VBP)

Humanities: Debates On Ethics of Big Data (DEBD)

Objective of the project

- Model, deploy and implement a Big Data Management backbone



Part 1 – The Landing Zone

- Identify data sources
- Implement data collectors
- Decide on the structure and deploy the temporal landing zone
  - This is required when pulling data from external sources
- Implement the Data Persistence Loaders per source
- Decide on the structure and deploy the Persistent Landing Zone

Part 2 – The Formatted and Explotaition Zone

- Decide on the kind of analytics/queries to be applied
  - Descriptive analytics
  - Predictive analytics

this will impact the choice of technology for the Formatted Zone

- Decide on the structure and deploy the Formatted Zone
- Implement the Data Formatters
  - Data cleaning
  - Data integration
  - Data reconciliation
- Decide on the structure and deploy the Explotation Zone

Closing