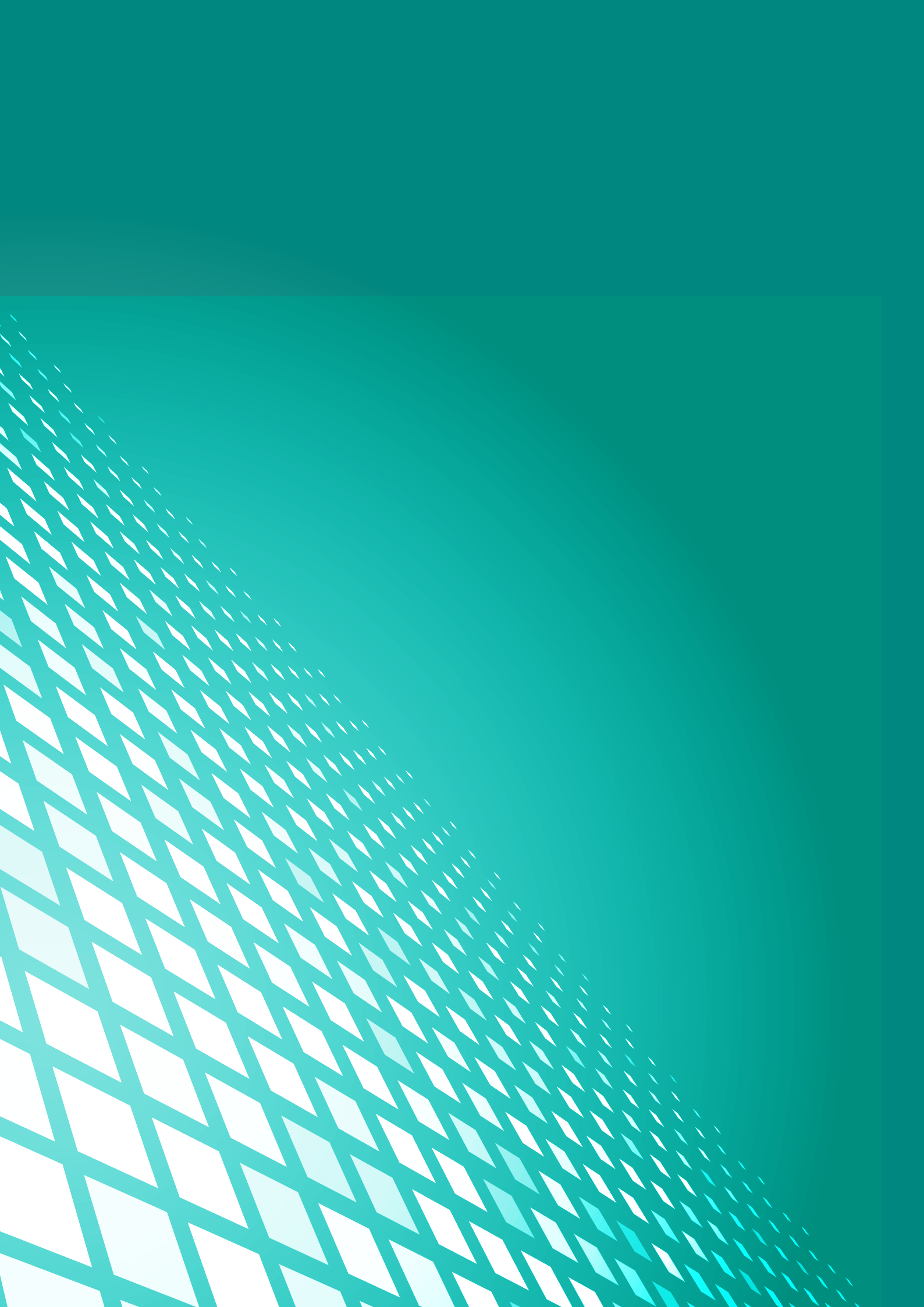
GUIDELINES FOR BEST PRACTICES FOR PUBLIC SECTOR'S DATA SCIENTISTS

THE GOVERNMENT
OF THE GRAND DUCHY OF LUXEMBOURG
Ministry for Digitalisation

THE GOVERNMENT
OF THE GRAND DUCHY OF LUXEMBOURG
Ministry of Research and Higher Education



Preamble

In the ever-evolving landscape of data science, navigating the complexities and ethical considerations inherent in harnessing data-driven insights is paramount. As the volume and velocity of data continue to surge, it becomes imperative to establish a robust framework that delineates the principles, objectives, and goals guiding our endeavours in this field and that any use will take place in compliance with the research ethics and the law.

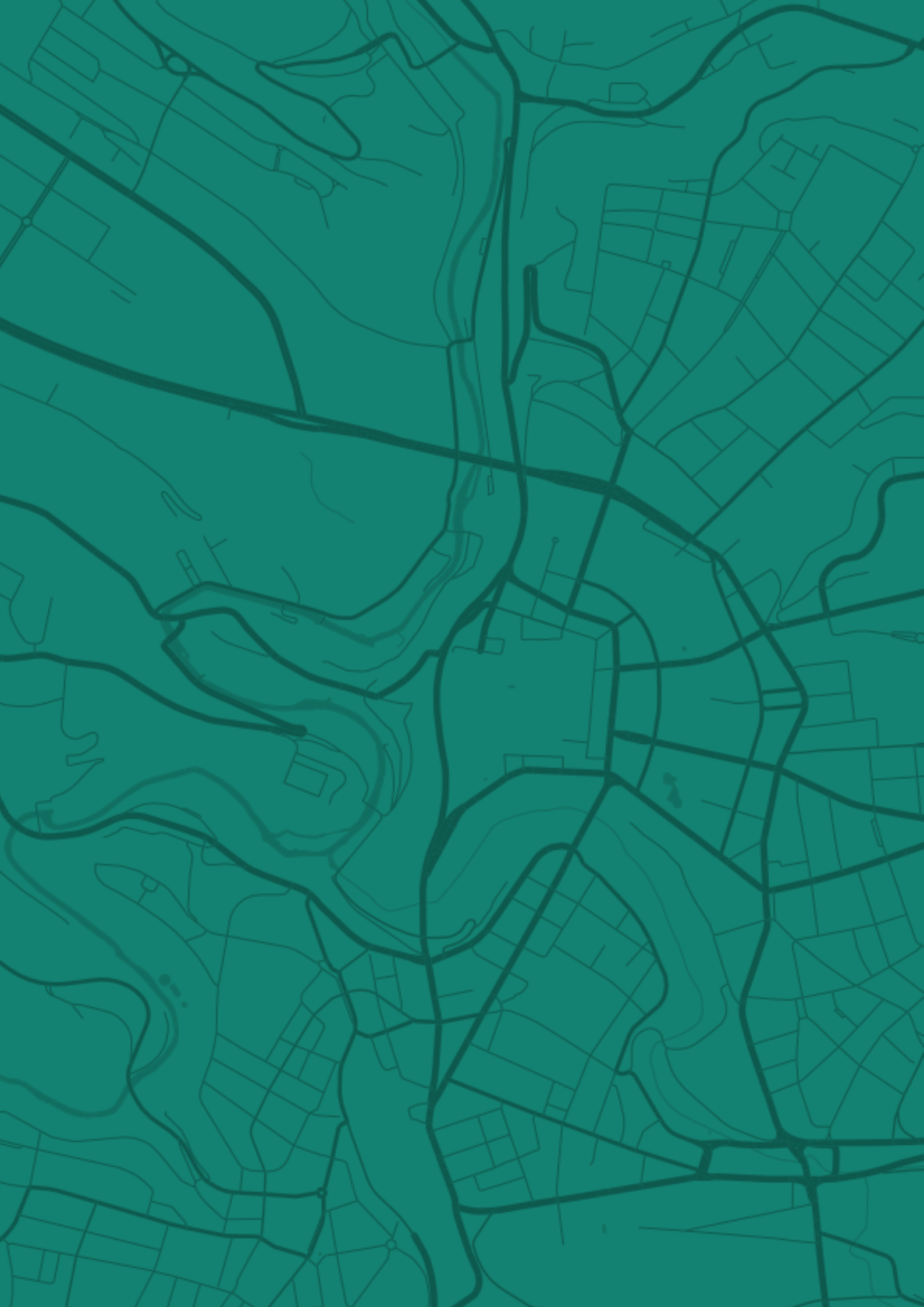
This document is tailored to meet the unique needs of data scientists, operating within the public sector in Luxembourg, and the current legislative and technical ecosystem. Its primary objective is to provide a clear and comprehensive framework that guides practitioners in navigating the technical, ethical, and societal challenges inherent in their work.

By promoting principles of integrity, reproducibility and maintainability, this guideline aims to empower public sector data scientists to harness the full potential of data science.

Through a commitment to continuous learning, collaboration, and adherence to best practices, we strive to cultivate a community of data scientists dedicated to advancing the public good through ethical and responsible data-driven decision-making.

The document provides both, a keyword driven glossary to find specific information quickly as well as deeper insights on all aspects of successful data science.

Section 1 introduces some definitions that are important to set the scope. Section 2 introduces general objectives that data scientists should strive for, and Section 3 continues with more specific best practices and guidelines. Finally, Sections 4 and 5 are more focused on mid to long term goals by introducing continuous monitoring and improvement as well as lifecycle management.



Contents

1. Definitions

Data scientists at work face a broad range of different concepts, that are the subject of this section. Their core expertise concern domains such as statistics, mathematics, machine learning and programming, while they are also confronted with concerns such as ethics, law, software engineering and project management. More to the ground, data scientists implement data pipelines to manipulate data to extract insights, reports and visual information. Additionally, should their work include machine learning tools they may also make use of off-the-shelf models or provide trained or refined models of their own.

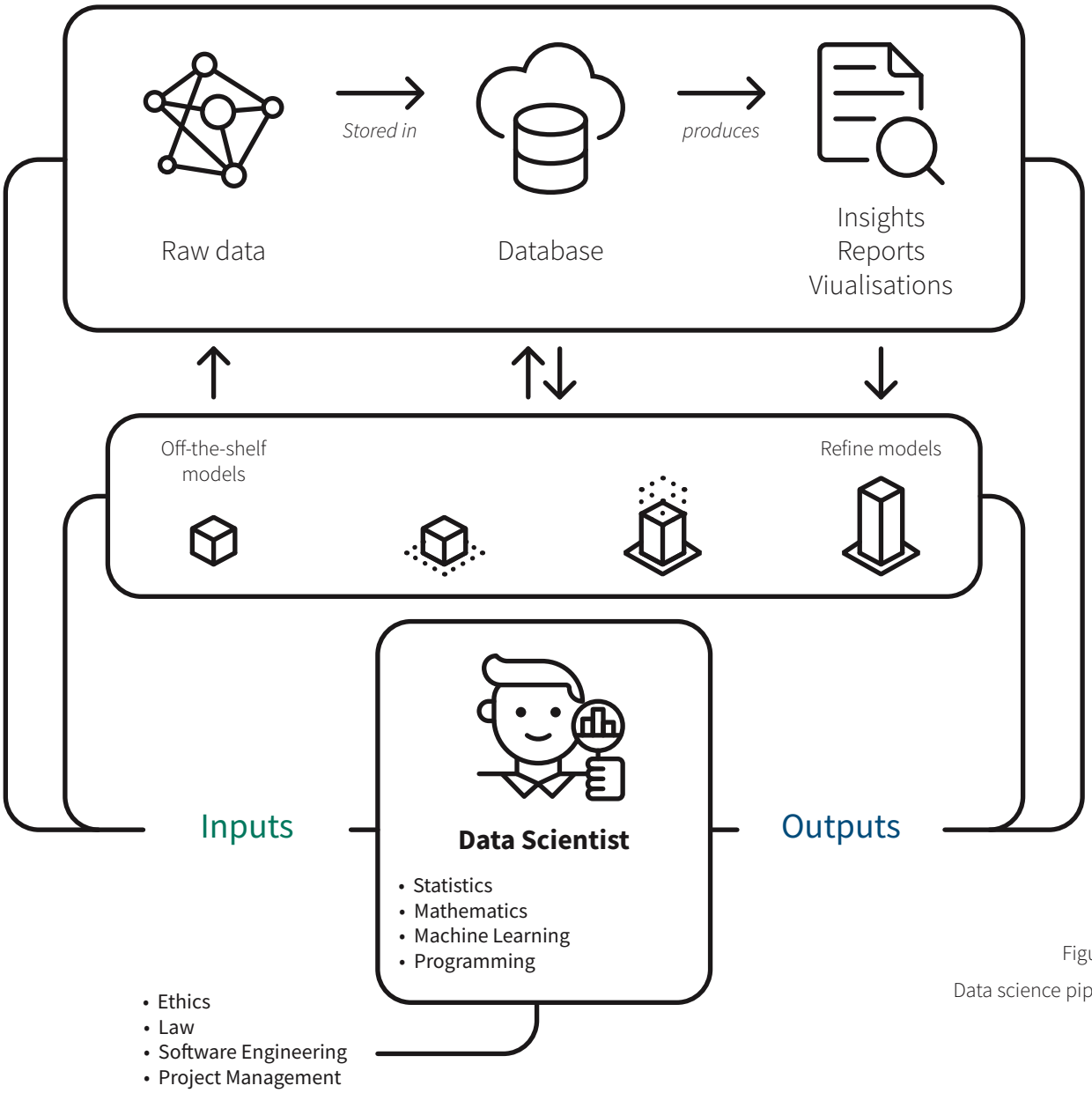


Figure 1:
Data science pipeline

- **Data:** A data point represents information, fact, statistics collected and stored for reference, analysis, or future use. It can exist in various forms, such as text, numbers, images, or multimedia, and serves as the foundation for analysis and decision-making in various fields. In some instances, computer code can also be considered data.
- **Dataset:** A dataset is a structured collection of related data points, or coming with metadata allowing for structuring, or information organized in a specific format, typically stored in databases or files. It serves as a cohesive unit for analysis, research, or problem-solving, containing multiple observations or instances along with their associated attributes or variables.
- **Metadata:** Metadata is additional information that provides context, structure and details about the underlying data or dataset and helps to facilitate the organization, management and understanding of the prior.
- **Data Analytics:** Data analytics involves the process of examining, cleansing, transforming, and interpreting data to extract valuable insights, patterns, or trends. It utilizes various tools, techniques, and statistical methods to uncover meaningful information that can aid in decision-making and strategy formulation.
- **Data Science:** Data science is an interdisciplinary field that encompasses methodologies, algorithms, and tools to extract insights or knowledge from data. It involves statistical analysis, machine learning, programming, and domain expertise to solve complex problems, develop predictive models, and derive actionable insights from datasets.
- **Data Scientist:** A data scientist is a professional skilled in handling and analysing large volumes of data using statistical analysis, programming, and domain knowledge. They leverage their expertise in data science techniques to explore data, develop models, and interpret findings to solve business problems or inform decision-making processes.
- **Data Science Pipeline:** A systematic and automated process that covers the collection, crawling, processing, analysis, and deployment of data to address data science specific problems. It orchestrates a sequence of connected stages to streamline and automate these tasks, thus providing data science projects with consistent, reproducible workflows.

- **Model:** In the context of data science, a model refers to a representation of a real-world system or phenomenon. They are an essential output of a data science pipeline and often the main motivation to even start a project. Models can also exist off-the-shelf and serve as input or a means of transforming data. The design and evolution of models over the span of a data science project is in close relation with the observations made during data analytics. Usually, one would start with a basic model and refine or retrain in iterations to create the final version.
- **Software engineering:** Software engineering is a discipline within computer science focused on the systematic design, development, testing, and maintenance of software systems. Some of the principles of software engineering also apply directly to data science (e.g. programming best practices). However, software engineering, as a discipline, should rather be seen as something that exists aside from data science but will have to be considered for project that go beyond data analytics and should provide a working software suite as result.

1.1. Data analytics types

Typically, a data scientist can perform six types of analysis:

- **Descriptive analytics:** Descriptive analytics involves describing a past event using techniques known as descriptive statistics. It helps in summarizing and presenting data to provide insights into patterns and trends. Typically, descriptive analyses help answer questions like "how many driving under the influence fines were issued daily during the past year?" There can be various ways to answer this question, such as providing a daily average of fines over the entire period, averaging for weekdays and weekends separately, or presenting a graph illustrating the daily total over the period. There isn't necessarily a single correct answer, and the response will depend on the type of data analysed, what aspects are to be highlighted (for instance, if the highlight should be on an increase in fines during weekends or even on seasonal fluctuations, then a graph might be more suitable), and who is asking the question.
- **Diagnostic analytics:** This type involves the examination of data to understand why certain events happened or what factors influenced those events. It helps in identifying correlations and causation.
- **Real-time analytics** involves the processing and analysis of data as it comes in, providing instantaneous insights. This approach allows for decision-making in real-time, reacting quickly to new conditions or events. A prominent use case for real-time analytics is the need to generate alerts under pre-defined or pre-identified conditions, which can then be integrated in existing alert systems or developed into live dedicated dashboards.
- **Predictive analytics:** Using the previous example, predictive analytics would address a question such as "how many fines can we expect during the week of the 15th of August?" The approach typically involves modelling time series based on past data, allowing predictions based on this model. However,

it's essential to be cautious here: predictive analytics cannot answer a similar question such as "how many fines can we expect during the week of the 15th of August if we install a radar at intersection ABC?" Predictive analytics often relies on "time series" data. Appropriate methods considering the specificities of time series (stationarity, seasonality, co-integration, etc.) must be employed.

- **Prescriptive analytics:** While predictive analytics recommends actions or strategies based on predictive models to optimize outcomes, prescriptive analytics provides guidance on what actions to take to achieve that desired outcome. Prescriptive analytics utilizes optimization and simulation techniques to suggest the best course of action based on the predictions made by predictive models. Example: Recommending marketing strategies based on predictive models to maximize sales.
- **Causal analytics:** Causal analytics aims to determine the impact of an intervention. There isn't a one-size-fits-all methodology here, but in general, it's crucial to determine what is referred to in the jargon as an "identification strategy." Ideally, this strategy involves an experiment. However, practical limitations, such as cost, ethics, or feasibility, might prevent its implementation. In the absence of an experiment, statistical approaches can approximate ideal experimental conditions to determine the causal impact of an intervention. Drawing a directed acyclic graph (DAG) to determine which variables should be considered or discarded in the identification strategy is highly recommended.

While descriptive analytics focuses on summarizing historical data, predictive analytics forecasts future events, prescriptive analytics suggests actions based on those forecasts, and causal analytics delves into understanding the relationships between causes and effects in data. Each type of analytics serves a different purpose and aids in decision-making in various domains, from business to scientific research.

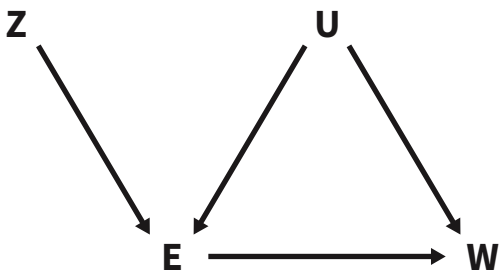


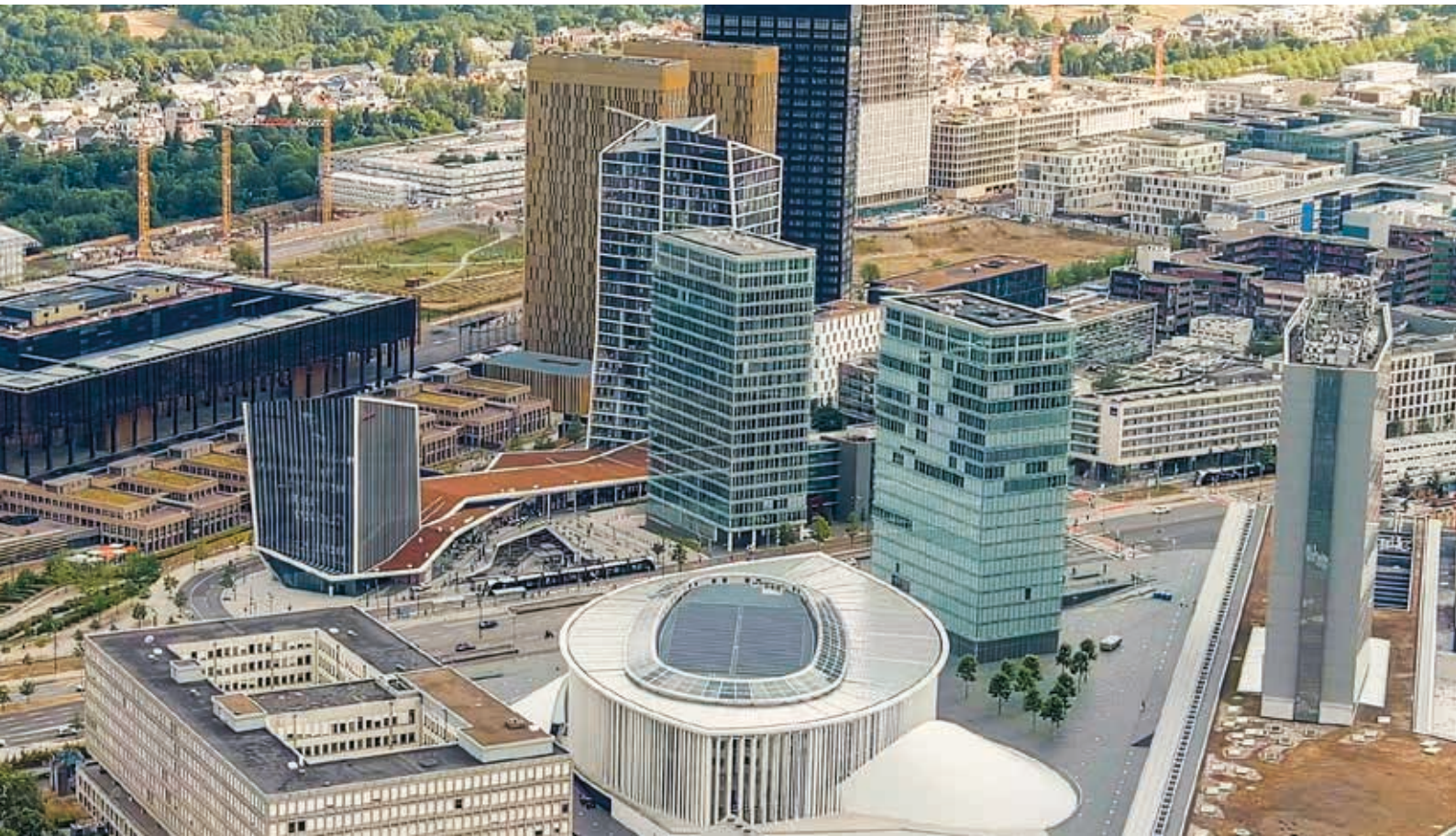
Figure 2: A DAG illustrating the "instrumental variables" approach for causal inference. Source: Statistical rethinking with brms, ggplot2 and the tidyverse [1]

1.2. Software and data engineering techniques

Various techniques exist to promote collaboration in data science projects and support their deployment and integration.

- **Continuous Integration (CI):** CI is a development practice where developers regularly merge their code changes into a shared repository. The goal is to detect integration issues early by automating builds and running tests, ensuring that new code additions seamlessly integrate with the existing codebase.
- **Continuous Delivery (CD):** CD extends CI by automatically deploying code changes to production or staging environments after passing the automated testing phase. It ensures that software can be reliably released at any time, reducing the manual intervention required for deployment.
- **Container (OCI - Open Container Initiative):** Containers are a form of lightweight, portable, and executable software packaging that encapsulates an application and its dependencies. OCI is a standard specification for container formats and runtimes, ensuring compatibility and interoperability across different container platforms.
- **DevOps:** DevOps is a cultural and technical approach that aims to bridge the gap between development (Dev) and operations (Ops) teams. It emphasizes collaboration, automation, and shared responsibility throughout the software development lifecycle to deliver high-quality software more rapidly and efficiently.

- **DataOps:** DataOps is a methodology that applies DevOps principles to data engineering and management. It focuses on streamlining and automating data workflows, ensuring collaboration between data teams, and improving the agility and quality of data-driven applications.
- **MLOps:** MLOps is a set of practices that combine machine learning (ML) and DevOps to enhance the lifecycle management of ML models. It involves automating the deployment, monitoring, and management of ML models at scale, ensuring their reliability, reproducibility, and continual improvement.
- **Software Testing:** Systematic and automated processes that verify functionality, detect defects and ensure requirements of a project:
  - **Unit testing:** Process of testing individual, small units or components in isolation to ensure their correctness and functionality, thus providing low-level, early bug detection.
  - **Functional testing:** Assessment of a project's features against predefined specifications about their behaviour and performance, thus ensuring the project functions as expected regarding specified requirements.
  - **Integration testing:** Evaluation of deployment in a production-like environment to prevent issues related to data flow and available resources.
  - **Regression testing:** Prevents recent changes to the codebase from introducing breaking changes in existing features, bugs in the project or qualitative degradations.



2. Core Aspects of Data Science

2.1. Understanding the domain

At its core, a data science project can be considered as the implementation of diverse analytical tools and practices toward solving a business problem, *i.e.* a task pertaining to a specific domain expertise. In this respect, aligning data science efforts with the domain expertise is a key aspect that covers every step of a successful data science project:

1. Relevance to set goals: grasping and keeping in mind the business problem ensures that the project will not deviate from its objectives.
2. Domain expertise: a deep understanding not only of the specific project's objective, but also of the broader field it pertains to, leads to better interpretation of preliminary result, and allows for more insightful tracks to be investigated.
3. Knowledge extraction and model development: identifying the features relevant to the business problem is of the utmost importance toward implementing both accurate and efficient models. Domain expertise may come with pre-identified correlations between features and goals.
4. Continuously challenging the requirements and developed solutions to make sure that the final delivery provides tangible benefits.

For these reasons, a data scientist should always strive, from the very beginning and all along the life of a project, to build and deepen their knowledge of the domain the project relates to.

2.2. Understanding and defining the project

For any project to be successful, it is of utmost importance to define the goals and objectives in a clear and tractable way. This is also true for data science projects. Training models without a defined way of evaluating performance will ultimately make the move into production impossible. At the start of a data science project the stakeholders should therefore:

1. Define software related requirements.
2. Decide if this is indeed a data science project, or if the expected goal can be reached without conventional software development. In the latter case there is no need to go to the next steps in this list.
3. Define model performance related requirements and metrics to prove them.
4. Select a data(sub)set to use for the evaluation.
5. Make sure that both training and evaluation data are representative of the production environment.
6. Iterate over the previous points several times throughout the project execution and refine the goals if necessary.

2.3. Legal and ethical aspects

Likewise, legal and ethical aspects are other crucial considerations that span the whole lifetime of a data science project. Indeed, all stages may be subject to legal regulations and compliance.

1. **Personal data processing:** the following principles should be observed:
    a. Clear definition of the purpose, which is the specific outcome on the concrete question that you want to answer and determination of the data that are needed to answer the question.
    b. Establishment of a legal basis.
    c. Information provision to data subjects.
    d. Documentation in the Processing Records.
    e. Compliance with the provisions in the law of 1 August 2018 on data protection.
2. **Data management:** All along the lifecycle of a project, legal and ethical aspects pervade the management of data:
    a. **Data collection and privacy laws:** data privacy and the legality of collecting specific types of data must be considered from the very beginning. This includes compliance with the General Data Protection Regulation [2] (GDPR) if data contains personal information. Additionally, stay alert to evolution in the legislation surrounding the Data Governance Act [3], the AI Act [4] and the Data Act [5]. If your project would require data from another public administration, know that access to data from another administration is not automatic and might not even be possible at all. Discuss the data needs as early as possible in the process and involve each of the administrations DPOs as early as possible. Also consider getting advice from the CGPD.
    b. **Data processing, storage and security:** legal requirements related to data security and protection do affect where and how data is stored and processed.
    c. **Project conclusion, data retention and disposal:** Legal consideration about data retention and the right to be forgotten must be diligently observed when concluding a project. Any data unnecessary to long-term production should then be appropriately discarded as a safeguard.
3. **Data exploration and fair use:** use of data for exploration or feature engineering must comply with legal and ethical standards. Beyond intellectual property rights, this also relates to more ethical concerns such as biases in data relating to actual persons and populations for instance.
4. **Model development, evaluation and fairness:** ethical and legal considerations regarding bias and non-discrimination go beyond data exploration and span over all the implementation of data analytics as a model, as well as its validation and evaluation.
5. **Deployment, integration and transparency:** certain domains and applications, especially in sectors such as finance and healthcare, come with strict regulatory compliance regarding transparency or explainability of provided applications.
6. **Long-term maintenance and compliance updates:** As legal frameworks evolve, a project in production must be monitored and updated to stay in compliance with these evolutions.

Towards these it is crucial in data analytics projects to ensure responsible and fair use of data. The following practices aim at achieving legally compliant and more ethical data usages:

- **Data anonymisation** refers to irreversibly de-identifying any personal data. This approach is recommended where personal data are not necessary to successfully execute a data project. However, irreversible anonymisation of personal data in the sense of the GDPR is challenging to achieve. Therefore, the feasibility of data anonymisation should be discussed with a qualified data protection expert, such as the Data Protection Officer (DPO) of the data scientist's organisation.
- **Informed consent:** under certain circumstances, it is a legal requirement to obtain an explicit informed consent from individuals (data subjects) when collecting and/or re-using personal data. When obtaining an informed consent, the information to be provided to the data subjects must stipulate how data will be used. The data subjects must be given a possibility to accept or refuse the use of their data for each specific purpose (for example, by ticking a box or through similar means).
- **Transparency** about data collection, storage and usage practices is a fundamental way of communicating unambiguously to individuals how their data will be used and who will have access to it.
- **Responsibility and professionalism** in handling data, in accordance with the accountable organisation’s policies and rules is of the utmost importance.
- **Avoiding harm** to individuals or communities must always be a concern when dealing with data that concern individuals, demographics, social matters, etc. It is there crucial to consider potential negative impacts and take steps to minimize risks.
- **Social and ethical implications** on broader societal considerations should go under scrutiny. Assess how your project might impact society, communities, or vulnerable groups.
- **Continuous Evaluation** allows data scientists to regularly assess the ethical implications of their data practices. Stay updated on evolving ethical standards and adapt your practices accordingly.

2.4. Data quality

Data quality refers to the adequation of data with the project's purpose. This encompasses different characteristics of the data:

- **FAIR:** Check that your data adheres to the Findable Accessible Interoperable Reusable principles [6].
- **Accuracy** measures how well the data stays true to the real-world information it conveys. This can be hindered for instance by badly calibrated sensors or measurement bugs.
- **Completeness,** or if all data necessary to the project is available. Missing data may reduce analysis quality or even render it impossible in practice.
- **Consistency,** or how data from different sources align in term of format, units, and in time.
- **Precision** refers to the level of detail in the data. It ensures that data elements are accurately represented without ambiguity.
- **Reliability** which indicates how data remains dependable and consistent over time. Unreliable data usually changes frequently or shows inconsistencies.
- **Relevance** to the task at hand, how much the data is directly applicable to the project's objectives, or requires either preprocessing, additional information, or both.
- **Timeliness,** or freshness of data, may concern more specifically the case where history data is expected to be and stay representative of unobserved data, and associated events, to come. In this context it is essential to be aware of notions of data staleness and deviation over time.
- **Validity** toward predefined rules or constraints. This broadly covers expected formats, numerical conditions for some algorithms to work in data analysis, or legal compliance.
- **Understandability** ensures that Data should be easily understood by its intended audience. Clear definitions, labels, and metadata contribute to better data comprehension.
- **Uniqueness** of data, to prevent unexpected duplicates degrading the soundness of data analysis.

2.5. Data science pipeline

A data science pipeline aims at structuring the whole process of a data science project as a succession of steps that all share common core concepts. The succession of these steps ultimately leads to the successful implementation of transforming raw data into the expected results. The steps in question vary from project to project, depending on the raw data, the results to produce, the approaches implemented, but usually come down to (part of) the following list:

1. Data preparation
2. Feature engineering
3. Modelling
4. Model training
5. Evaluation
6. Inference
7. Interpretation

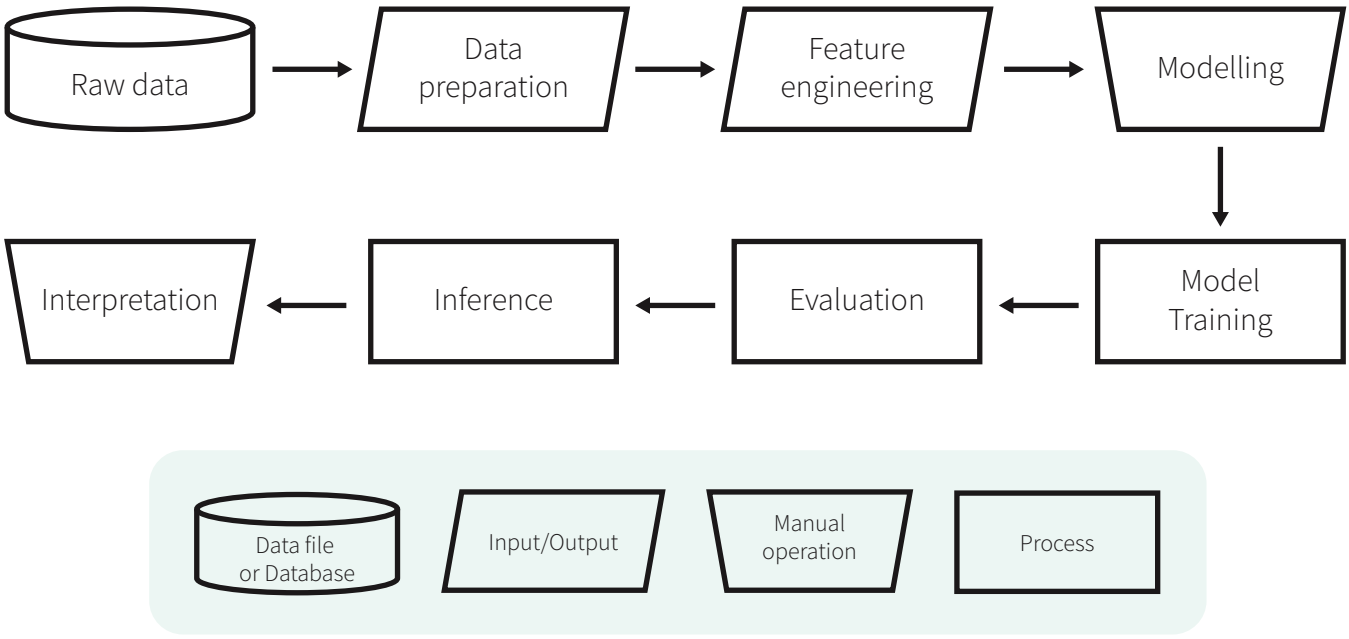


Figure 3:
Flowchart [7] of a data science pipeline

This document does not aim to describe these in detail, but rather to provide good practices in how to implement them as reproducible, interchangeable steps of a data science pipeline. In fact, all these steps can be decomposed into three stages:

1. Data collection: data necessary to the current step is retrieved from its source.
2. Data transformation: input data is parsed to produce output data, whether it is tidy data after raw data cleaning, machine learning model trained from a training subset of all data, etc.
3. Data storage: output data is stored for further usage (by next steps or for integration in production).

These considerations highlight that all steps rely on mainly the same two technical needs to run:

1. Data storage for both input data and output data.
2. Execution environment for implementation of data transformations.

To minimize efforts in providing such solutions as well as systematize deployments, ideally any data science project must rely on a single, standard solution for each of these two aspects. With a single solution for data storage, it becomes possible to focus on the peculiarities of the data at each step of the pipeline. The execution (or compute) environment should make it possible to ignore the underlying hardware completely and concentrate on the code only. Standard containerization as defined by the OCI initiative is usually the way to go, as this eases a lot both maintainability and reproducibility of projects, all along their conception and beyond.

3. Best practices and guidelines

3.1. Data science pipeline

Data scientists should know the current IT and legal ecosystem. The main provider of tools and infrastructure for the public sector is the CTIE, so if you require anything IT-related, reach out to them[1]. Depending on your administration, the provider might instead be the CGIE, which is the main provider for the Ministry of Education, or the SIGI, in the case of municipalities. In any case, reach out to your IT representatives/department first, and see what they have to offer to support you.

Data Scientists should be aware of the other main players in the data ecosystem: the public entities themselves, as data holders and data processors, the Luxembourg National Data Service (LNDS) as enabler for data exchanges, the CNPD (national data protection authority) and its public sector counterpart CGPD, the cybersecurity actors like GOVCERT (as data needs to be protected), or the Open Data Portal as a free-of-charge and ready-to-use data provider. The STATEC, as the national statistical institute is another important actor, and it is important to get to know their business and them.

To get to know the data and IT ecosystem better, consider joining the Data Science and Business Intelligence meetups. These groups meet frequently and representatives of all the above-mentioned players join the meetings to discuss projects, ideas, and generally to get to know one another. Contact us at datascience@digital.etat.lu to join these groups.

Every year, the Ministry for Digitalisation organises two calls for projects, the DATA4GOV and the AI4GOV, which are great opportunities for data scientists to submit project ideas that may go beyond the daily business of their respective administrations. Selected projects get expertise and support from the Ministry for Digitalisation.

1 You may check the LogON platform

3.2. Bias

Data analysis biases are systemic errors that can occur either in data collection, analysis, interpretation or publication. Leading to inaccurate or misleading conclusions, it is important to identify the different kinds of biases to recognize and prevent them:

- **Selection bias** may occur when an analysis that aims to relate to a broad population must deal with only a subset of it, usually because the size of the total population makes it impractical to either collect data that pertains to the whole population or to implement approaches to the scale of it. The bias here is introduced when the selection does not ensure the part is representative of the whole. From there on, conclusions will rely on a false assumption, resulting in skewed data that reflect the undetected skew in data representativity. Selection bias can also be introduced as the data cleaning step, during which applying certain filters to the data could result in a non-random sample of the population. Comparing descriptive statistics generated on the sample with available aggregate statistics of the population can help detect selection bias.
- **Survivorship bias** appears when only a portion of what is to be represented made it to the data collection, often keeping only well-structured, well-formatted data, thus on the assumption that data that don't comply to expectations cannot carry extractable information anymore. While essentially true, this assumption disregards the fact that garbage data are informative in themselves (*e.g.* degraded data from a sensor says a lot about the health of the sensor or the network it sends information through), but also leads to an overestimation of fault-free behaviours in the systems data relate to.

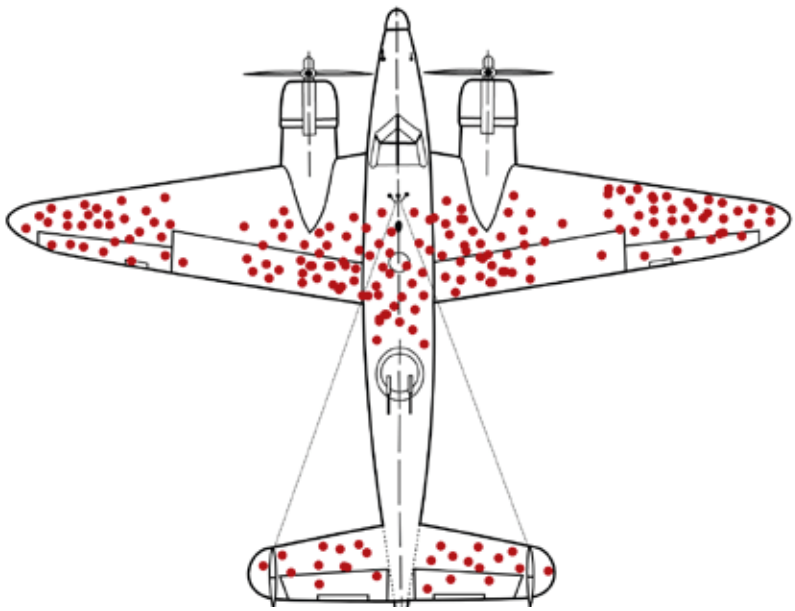


Figure 4 Location of the hits in the aircraft. McGeddon, CC BY-SA 4.0, via Wikimedia Commons. Illustration of survivorship bias, inspired by the seminal paper of Abraham Wald "A method of estimating plane vulnerability based on damage of survivors" (1943) [8]

- **Confirmation bias / observer bias** refer to the inclination to favour data that confirm pre-existing beliefs or validate hypotheses. This leads to an oriented, incomplete interpretation of data that only contributes to flourishing itself.
- **Sampling bias** occurs when data analysis is lead only on a sample of the data. Even from representative data can be sampled an unrepresentative, bias subset of it. This leads to ungeneralizable results and conclusions, that only apply to the biased sample.
- **Outlier bias** can appear when data contains a portion of observations that a) differ so much from normal data that they remain a significant outlier all along the analysis; and b) represent a part of data significantly larger than their part of the general population. This leads to a significant skew in analysis, where the outliers hinder the analysis of norms and trends in the data.
- **Reporting bias / publication bias** refer to the tendency to identify, select or publish results only significant, perhaps only positive regarding set objectives. This gives a distorted view of the data, misrepresents the true findings and overestimates the underlying effects.

In addition to hindering the quality of a data science project, these biases also introduce the risk to propagate, through publication and decision-making based on them. For instance, skews in any part of a data science analysis focused on individuals may lead to unfair, perhaps discriminatory conclusions when generalized to the entire population.

3.3. Security and Compliance

Ensuring data security and compliance with regulations is important, especially when dealing with sensitive or personal information.

GDPR (General Data Protection Regulation) defines the notion of privacy by design [9] as "*data protection through technology design*". With its article 25, "Data protection by design and by default," GDPR mandates organizations to integrate protection measures by technological means throughout the whole data processing lifecycle. Accent is put on personal data, and the obligation to limit it to what is strictly necessary for the project, in term of collection, processing, retention and accessibility.

Beyond GDPR and when personal data is essential to a solution, the concept of secure-by-design provides guidelines for robust security measures to implement from the very beginning of a project. Key measures are encryption, stringent access controls and regular security assessments. These ensure that personal data incorporated in a deployed solution is handled with maximum security, thus preventing potential risks and keeping processing of personal data compliant with GDPR.

In a broader consideration of compliance with laws, a data scientist should have the skills to discern sensitive data, the reason why it is sensitive (privacy, intellectual property, governmental or European secrecy, etc.). Getting acquainted with the juridical knowledge that concerns these reasons helps to adhere to broader legal obligations and ethical considerations.

Finally, maintaining a legislative watch is paramount for staying up to date of evolving requirements, whether they come from GDPR or other legal apparatus. This applies not only to the development and production parts of a project, but also to the stored data, that may remain stored for later uses, especially when trying to ensure reproducibility. This proactive approach allows to keep projects aligned with the latest legal standards, protect individuals' privacy and observe ethical standards, promoting responsibility and discipline among the data science community toward these.

3.4. Knowledge extraction

Knowledge extraction is the process of extracting meaningful insights, identifying patterns or behaviours from raw data through analysis and modelling techniques.

- **Understand the problem domain:** Gain a deep understanding of the problem you're solving and the domain-specific knowledge. Striving for domain expertise mitigates risks of deviating from the project's goals, guiding data analysis with deeper understanding of variables' real-world meaning and their interactions.

- **Exploratory Data Analysis (EDA)** allows for a better understanding of the relationships between variables and identify patterns or correlations in the data. It involves extracting from data statistics, visualisations of patterns, identifying and cleaning inconsistencies, and establishing relationships between variables. With techniques able to reveal correlations between variables, trends in data, EDA allows a data scientist to get better insights and formulate hypotheses toward further analytics and models development.
- **Handle missing values:** The absence of data conveys its own information; thus, it is paramount to address them in one way or another. One should understand why these values are missing: if the data generating process is tied to a business process, talk to the business. There might be very good reason why these data are missing, and trying to impute or discarding these observations might be nonsensical. Imputation using simple statistics should be avoided, as it could introduce statistical bias (mean-imputation is only advisable when data is missing completely at random, which is very rarely the case). Ideally, one should use predictive methods for imputing data, and do so using multiple imputation to account for the uncertainty introduced through the imputation procedure itself. Consider looking into tools to impute data such as sklearn [10] for Python or mice [11] for R.
- **Encode categorical variables:** One may need to convert categorical variables into numerical representations using techniques like one-hot encoding, label encoding, or target encoding. Explore advanced encoding methods like frequency encoding or mean encoding for high cardinality categorical variables.
- **Detecting and handling outliers:** To detect outliers it is first necessary to profile the data. Common approaches focus on determining the underlying distribution which is characterized by the mean and standard deviation. This information can then be used to determine outliers that reside on the edges of the distribution. Dealing then with outliers may yield techniques common with the handling of missing values. Outliers might also be a clue that the population is ill-defined: if you are analysing data on houses, but have a house with 20 bedrooms, then it is likely that this “house” is a “hotel”. Make sure that you are working with a well-defined, representative sample of the population you wish to study. Take time to consider if outliers truly are outliers. For instance, if the data-generating process is assumed to follow a power law, then very high, unusual (but rare) values might be completely expected (book sales, death tool of natural catastrophes, etc.).
- **Text data processing** aims at extracting data from texts to be fed into numerical tools, such as statistics or training of machine learning models dedicated to numerical data. This comprises not only tokenization or stemming for natural language processing tasks, but also analyses such as TF-IDF [12] to extract statistics about usage of each specific word in the text data.

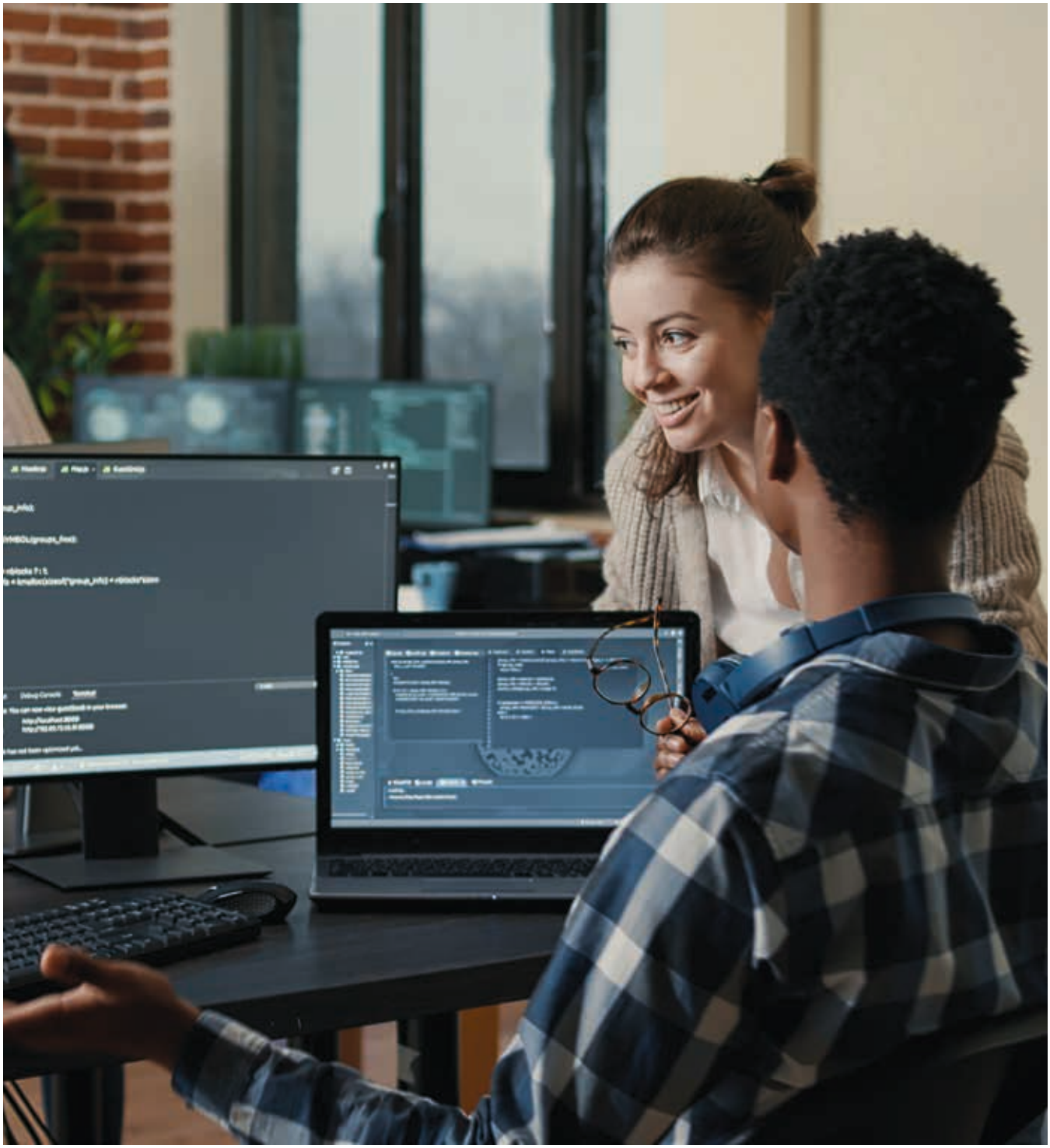
3.5. Data visualisation

Data visualisation in data science faces presenting complex information visually to convey meaningful insights. Effective data visualisation eases communication to a broad spectrum of audiences, thus aims at making patterns, behaviours, and relationships in data more understandable. Chenxin Li provides a thorough, though opinionated list of more technical guidelines [13].

- **Know Your Audience:** Understand who will be viewing your visualizations and tailor them to their level of expertise and specific needs. Design with clarity for the intended audience.
- **Choose the Right Chart Types:** Select appropriate chart types that effectively represent the relationships or patterns in your data. Bar charts, line charts, scatter plots, histograms, etc., serve different purposes.
- **Label and Annotate:** Label axes, data points, and provide clear titles and legends to convey context and aid interpretation. Use annotations to highlight key findings or trends.
- **Simplify and Declutter:** Avoid visual clutter by removing unnecessary elements. Emphasize the most critical information and simplify the design for better comprehension.
- **Use Colour Thoughtfully:** Use colour purposefully to highlight important data points or trends. Maintain consistency and ensure readability, considering colourblind-friendly palettes.
- **Provide Context:** Add context to your visualizations by including captions, descriptions, or summaries that explain the significance of the data and the insights derived.
- **Ensure Readability and Accessibility:** Use appropriate font sizes, legible text, and sufficient contrast for easy reading. Make visualizations accessible to all audiences, including those with disabilities.
- **Responsive Design and Interactivity:** Design visualizations that are responsive and adaptable to different devices. Use interactive elements (if applicable) to allow users to explore the data further.
- **Validate and Test:** Test your visualizations with a sample of your intended audience to gather feedback and ensure they effectively communicate the intended message.
- **Emphasize Storytelling:** Create a narrative flow in your visualizations to tell a story. Guide the audience through the data by presenting a logical sequence of information.
- **Continuous Improvement:** Learn from feedback and iterate on your visualizations. Strive for continuous improvement in design and effectiveness based on user responses and evolving needs.

3.6. Machine Learning

Machine learning refers to an ensemble of algorithmic and statistical techniques that are used automatically infer relationships, patterns within data. This training process produces an inference model that can be used to infer, predict observations, events or behaviours based on new data complying, in format, to the ones observed during training.



- **Model Hypothesis** relates to what data represents, as it is the model of patterns observed in data that is learned during the machine learning process. As it relies on the result of data discovery, crawling and knowledge extraction, the hypothesis trains on parts of the data identified as input features, that come from real-world observations or pre-analyses based on them, and other parts treated as output labels, or conclusions to draw from the features.
- **Data preparation** involves experimentation and validation to understand the data and prepare features and labels for model training. In addition to knowledge extraction techniques, various machine learning specific approaches may be implemented, depending on the data and the project goals:
  - **Transform Numerical Variables:** Consider applying transformations (logarithmic, square root, etc.) to numerical variables to make their distributions more Gaussian or to reduce skewness. This might even be mandatory when some layers of the model require non-zero data exclusively.
  - **Normalize or standardize numerical** features if it is necessary to ensure they are on a similar scale. This closely relates with numerical transformation and is mandatory for a lot of techniques to have any chance of converging when they heavily rely on all features to be comparable in the feature space.
  - **Time and date features:** Usually timestamps either come as machine timestamps or any variant of an ISO 8601 formatted timestamp. While both convey unambiguous information, it is not unusual for temporal patterns and behaviours to be characterize by more cyclic, seasonal concerns, like the hour of the day, the month of the year, etc. Preparing time and date features toward consists of extracting relevant information from them, such as the day of the week, month, season, or time difference between events, thus handling the cyclical nature of these features.
  - **Handling Class Imbalance (for Classification):** while it is not always advisable to handle class imbalance in data preparation [14], as class imbalance can be inherently of importance when training a classifier (oversampling a rare class may lead to over-predict it), there are some cases where it may be of interest. For instance, a machine learning model that would be trained to predict natural catastrophes is extremely likely to learn to never predict any and still achieve extremely high accuracy. Toward this, handling class imbalance involved techniques like oversampling, undersampling, or employing weighted loss functions to ensure fair model training [15]. But one should also look critically at the problem at hand: is it worth the effort to predict very rare events, or might it be better to invest time in trying to prepare and mitigate when the catastrophe hits?

- **Train/validation/test split:** Training machine learning models needs at least two sets of data that must never overlap to prevent data leakage. Under the hood, the model refines its internal parameters based on correlations discovered between features and labels in the training set. The secondary set is then used to measure how well these parameters still convey the same correlations on unseen examples and is also used to refine hyper parameters (e.g. model architecture or training procedure details). If possible, a third distinct set should be prepared by a data scientist to ensure that the learned parameters and hyper parameters generalize well on unknown data. Usually, the secondary set is called validation set and the tertiary set called test set, however, these two terms are occasionally swapped. Indeed, since both primary (training) and secondary (validation) sets are used during training, both may introduce overfit or biases inherent to those sets. Ensuring the model still works well on a third, independent set allows to mitigate this risk. Considering the tertiary set aims at preventing overfitting on secondary set, ideally it should be owned by a third party. This ensures that no decision can be made that, in favouring compliance with the tertiary set, would thus introduce an undesirable bias. The train/validation/test split should be done before anything else is done on the data to avoid data leakage: for example, feature engineering should be done after the split, and by part of the cross-validation loop, meaning all the feature engineering steps must be done on each separate fold.

- **Cross validation:** Cross validation (CV) expands on the idea of splitting data before training [16]. Instead of a single stationary split, it assesses model generalization on unseen data by validating the model on multiple splits of the same data into primary and secondary sets. Feature engineering should be done inside the CV-loop to avoid leakage. CV is also a good option in case that the dataset is too small to create the priorly suggested train/validation/test split. Two specific kinds of data where one should be cautious with cross-validation are temporal data and spatial data. As usual cross-validation techniques rely on splitting single datapoints between training and test sets regardless of their context, this leads to loss of temporal or spatial collocation of data. Regrouping data in time window of spatial blocks (*e.g.*, blockCV) before cross-validation is mandatory for the approach to be of any interest.
- **Hyperparameter Tuning** refers to the optimization of a model's external configuration settings that are not learned nor changed during training but may increase or decrease its performance significantly, such as the cross-validation parameters, the data split, the learning rate, etc. It relies on techniques like grid search, random search, or Bayesian optimization to find the best combination for improved performance. The objective is to fine-tune hyperparameters to optimize accuracy and generalization capacities of the model, which can be measured through the training results as well as running it on the validation set. It is important to note that this tuning should not happen too soon in a data science project, and that skipping it altogether can be considered if results and performance are up to expectations. Simple models and hyperparameters make for easier reproducibility, maintainability, and extensibility.

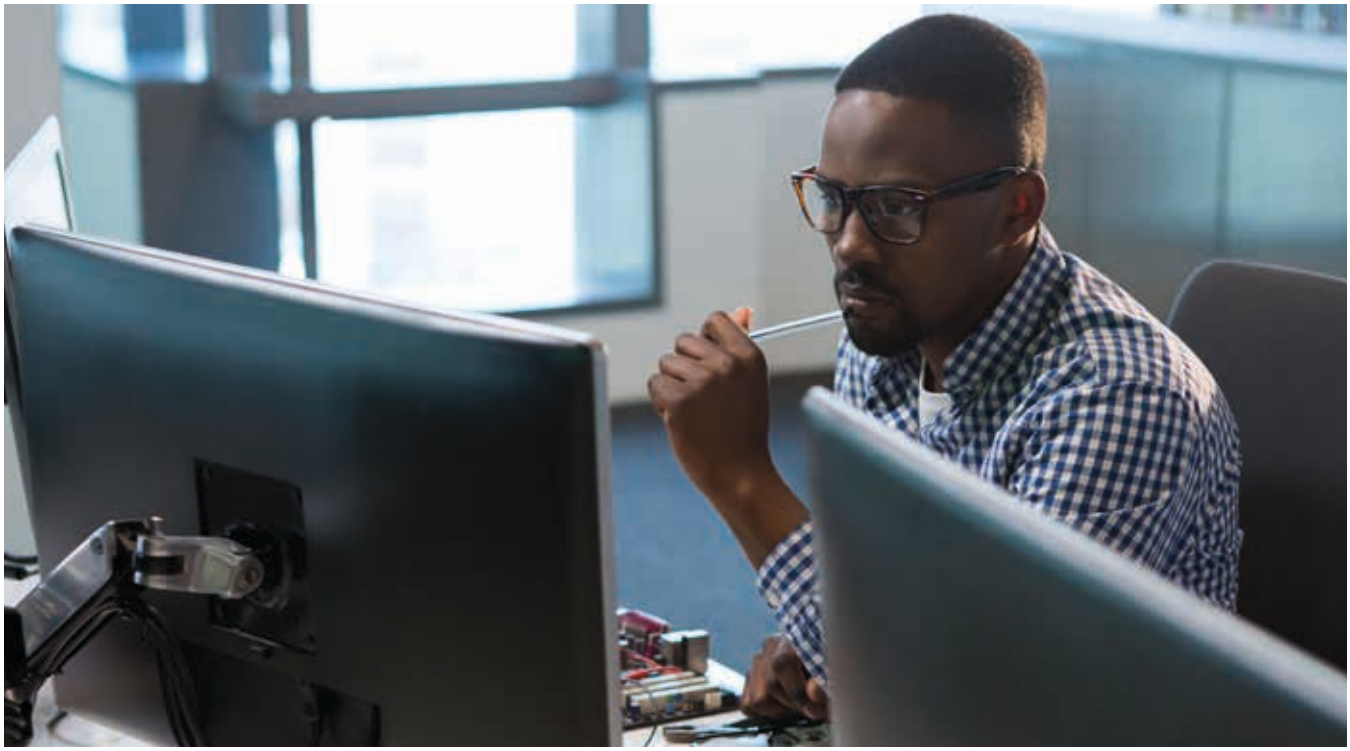
- **Fixing the random seed** for all machine learning tasks is mandatory to ensure reproducibility of results. By ensuring all pseudo-random parts of machine learning algorithm (initialization of learnable parameters, the data split, etc.) follow the same generation from one run to another allows to a) reproduce past experiments exactly and get the exact same results; and b) work on manual parameters or data preparation and compare results that can only differ because of these changes. Nevertheless, keeping to a single seed comes with the risk accidentally using a seed that significantly favours the training. Analysing how changing the seed affects the result in a good practice to circumvent this. Additionally, if changing between seeds systematically affects results, this can even question the approach or model used altogether. However, this analysis must not turn into seed fishing, where one would ultimately pick the seed that favours the training toward the expected results, thus introducing a confirmation bias in the data pipeline. Instead, it is a good practice to perform the exact same experiments on multiple random seeds and report the average performance, whenever computationally feasible.
- **Model Selection:** Choose appropriate algorithms or models based on the problem at hand and the characteristics of the data. Experiment with multiple models and compare their performance using cross-validation techniques. While having little to no effect between models that may use highly different internal architectures, a fix random seed for splitting the data remains of high importance to ensure the validity of comparing models' performances.
- **Regularization:** Use of regularization techniques (*e.g.*, L1/L2 regularization) allows to prevent overfitting and underfitting by containing or smoothing the learned parameters and penalizing complex models., ultimately enhancing the model's ability to handle new, unseen data.
- **Ensemble Methods:** Consider using ensemble methods (*e.g.*, bagging, boosting, stacking) to combine multiple models for improved performance and robustness.
- **Model Interpretability:** When possible, strive for model interpretability, especially in critical decision-making scenarios. Understand how the model makes predictions to gain insights and trust. Understanding how models arrive at their predictions or decisions is increasingly important, especially in scenarios requiring interpretability or regulatory compliance.

3.7. Software Engineering

Nowadays data science encompasses a range of software engineering practices to ensure efficient and successful development, collaboration, and maintenance:

- **Version control (for the code):** Use version control systems (e.g., git [17]) to manage changes to your code, models, and data. This facilitates collaboration, tracks changes, and helps in reverting to previous versions if needed.
    - Branches allow data scientist to work on experiments, features or fixes in a codebase isolated from the main one. It facilitates both parallel development through forking from the main branch and collaboration through merges back into the main branch. Branches are however a two-edged sword, with the risk of stale branches going under the radar and cluttering the codebase. To this regard, periodical pruning of stale branches streamlines branch management and helps to keep only relevant branches alive. Consider adopting trunk-based development [18] if you work in a team or a project.
    - Additionally, .gitignore [19] files help to ensure files that must not be committed, whether for size, type, legal or ethical reasons, won't be.

- **Unit testing** aims at validating small, individual portions of code. By ensuring reliability of functions, unit tests help preventing low-level errors and enhance the robustness of algorithms underlying data analyses.
- **Code reviews** must be led by a third person and aim at inspecting the code to estimate its readability, ensure the implementation of best practices, detect errors and maintain an overall high code quality. Peer feedback, in addition to follow best practices standards and ensure code robustness, also allows to spark new ideas and tracks to follow.
- **Documentation** is a key aspect of collaboration and maintainability. Toward this, the use of documentation generation tools (e.g., sphinx [20], doxygen [21], roxygen [22], etc.) allows to collocate code and documentation in the same file. For instance, class and function descriptions written in comments can be used toward generating documentation.
- **Collaborative platforms** (*e.g.*, GitHub [23], GitLab [24], Bitbucket [25], etc.) allow for teamwork by providing not only version control, but also issue tracking, code review features and integrated documentation.
- **CI/CD pipelines,** nowadays integrated in all collaborative platforms, allow for test and deployment automation, speeding up such processes all along the lifecycle of a data science project. They also systemize tests and deployments, mitigating the risk of introducing errors.
- **Testing data pipelines** is crucial to ensure data processing reliability, in term of stability, robustness, features and results. For instance, one could use testthat [26] for R or pytest [27] for Python. If possible, execute tests after each push to a collaborative platform using automated runners.



3.8. Reproducibility

The ability to replicate and verify analyses is crucial for ensuring the validity and reliability of results, making it a key practice in data science. For a data science project, this consists of the possibility, for anyone with the right credentials, to be able to:

1. Access at least raw, maybe prepared, tidy data
2. Access data pipelines
3. Execute data pipelines on raw or tidy data
4. Replicate and compare results

These must be ensured as early as possible and throughout the lifecycle of a project and remain true afterwards. Several solutions and practices exist and must be used, implemented, toward reproducibility:

- **Plain text scripts:** Write your analysis in plain text scripts, namely .py or .R files for Python and R respectively. Plain text scripts are production-ready by nature, as they can be usually deployed as is. Interactive tools, such as *e.g.* Jupyter Notebooks, should be avoided. The interactive tools might be helpful during early-stage data exploration, but they should be clearly separated from the project's data science pipeline and only be used by individuals and hence not be committed for collaboration. No-code tools should also be avoided.
- **Timestamping the training data:** Fix the training and testing datasets to be able to reproduce the same experiments.
- **Timestamping the used and trained models** for machine learning to improve trackability and the model evolution over the span of the project.
- **Saving training parameters with trained models.** Use human-readable, machine-parsable formats to store, update and access said parameters, such as JSON or YAML.
- **Fixing the versions of the software and libraries used: never** use "latest" tag, instead use fixed version i.e. "==3.2.1". Also, versions of software (*e.g.* Python 3.9.17) must be unambiguously specified in instructions (readme). It is also highly recommended to make use of individual virtual environments for each project. For Python *pyenv* [28] is commonly used to handle various versions of Python as well as virtual environments within each Python version. For R, using the ***renv package*** [29] is also highly recommended to at least fix package versions by generating a lock file.
- **Transparency:** Providing clear explanations and documentation of methodologies, hypothesis, assumptions, and limitations fosters trust, understanding, and informed decision-making.
- **Provide clear instructions** (README file) on how to run the code step by step. Though some data analytics might be complex thus hard to master, they must **never** be hard to run and reproduce.

- **Containerize your analysis:** When appropriate, provide your analysis in a containerized, deterministic and reproducible manner (e.g., Docker [30], Kubernetes [31]). Make sure that the same image gets built when building the container by using digests of base images rather than tags. Use lock files (for example, using the *pip freeze* command if you are using pyenv or a renv.lock file if you're using renv in R) to install the dependencies at build time. When in doubt, privileging solutions that comply with the Open Container Initiative [32] is a prudent choice towards long-term maintainability. External references, secrets and credentials **must** be kept out of the codebase, as well any built container image.
- **Automate**, when possible, as much as possible to remove human errors. Automation will enhance speed and accuracy of the analysis, through the use of build and workflow automation tools such as Snakemake [33] or Nextflow [34], for instance.

3.9. Maintainability

Well-structured, readable, and documented code ensures that analyses can be easily maintained, scaled, and improved upon, contributing to the longevity of a project.

- **Starting new projects with the newest libraries possible** ensures access to up-to-date, optimized features. It also mitigates the risk of relying on unsupported versions. Reversely, stale libraries (with no update for a long time) with no explicit support information should be treated cautiously when integrated in a project, and simply avoided at best.
- **Keep updating libraries versions,** when possible (dependencies, vulnerabilities), but make sure that you have enough unit tests to ensure that the code keeps producing the same results with these newer libraries.
- **Consider refactoring your code to improve it.** Improvements cover readability, efficiency, error management, fixing potential vulnerabilities, and keeping up to date with evolving project's requirements.
- **Keep updating software,** but make sure you can rollback in case of problems.
- **Keep updating models** with new data to address model drift. As data evolves over time, models trained during development will ultimately face data not entirely compliant with their inherent assumptions and requirements. Retraining at long-term intervals allows not only to keep a project to maintain alive, but also to realign deployed models so they stay true to their goals and efficiency.
- **Keep updating datasets** is crucial in addressing data drift and is to implement hand in hand with the previous practice. Indeed, including new data allows for catching up with changes in behaviours or patterns in the data, thus realigning accuracy and representativity of retrained models.
- **Use the recommended programming style for your language of choice.** For R, try to follow Google's [35] or Posit's [36], for Python, follow PEP 8 [37].

It is recommended to define and adhere to general project structure within an organization. The following illustrates an exemplary structure which can be used for guidance. In the below example and if you collaborate using git, you should only commit root level files as well as the contents of the src folder and include the other directories in your .gitignore file.

Figure 5: Example of a data science project structure

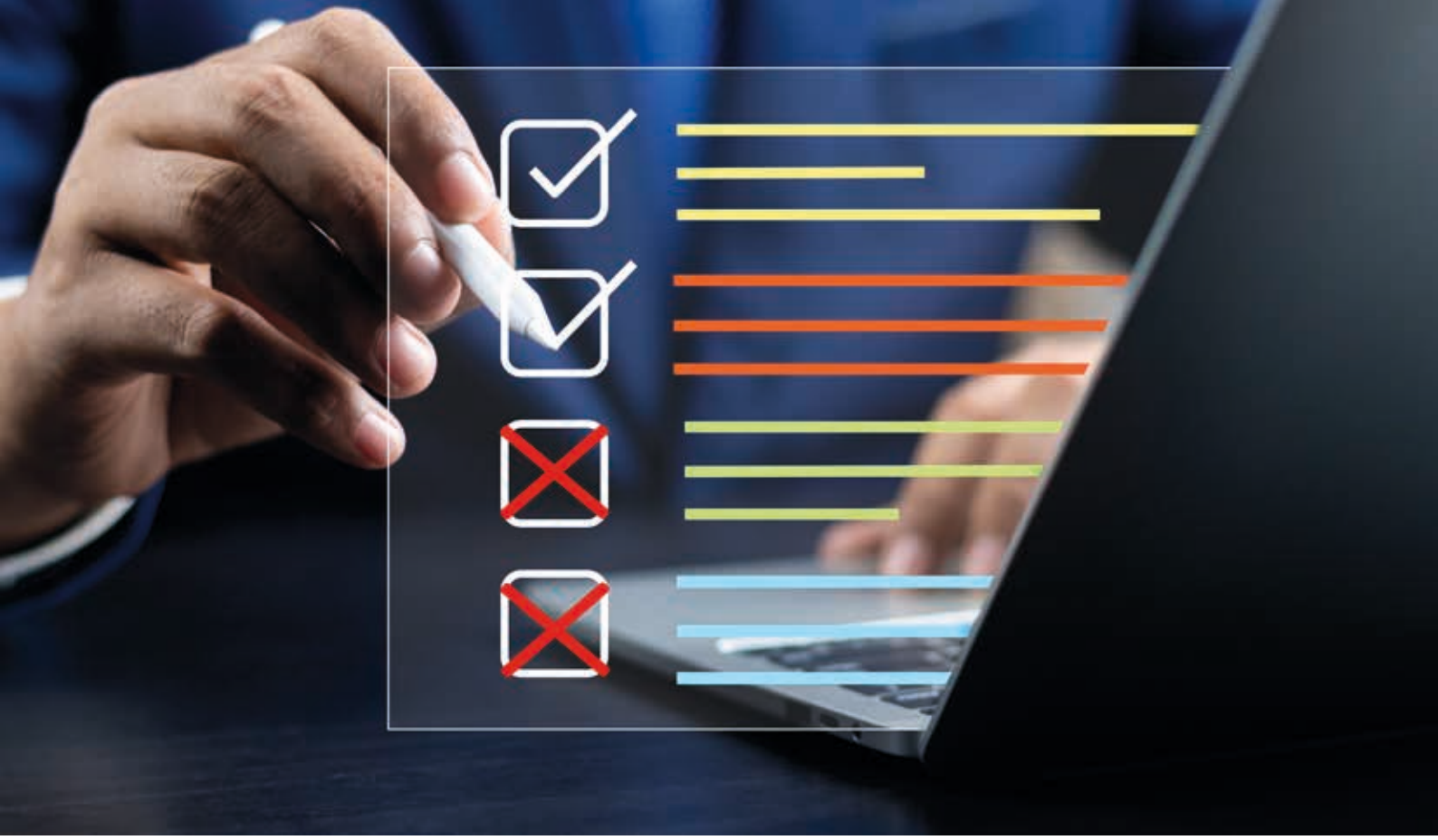
```
.                       # root level
├── .dockerignore       # excludes files from Dockerfile's COPY and ADD commands
├── .gitignore          # excludes most contents from commits (except root files and src)
├── Dockerfile          # script file to build or run the project in a container
├── README.md           # introduces the project and how to reproduce it
├── data
│   ├── external        # data from third party
│   ├── raw             # unprocessed data, should suffice to reproduce the project
│   └── processed       # output of data processing pipelines
├── exploration         # scripts and notebooks that are not meant for collaboration here
├── models              # trained and serialized models to be used and tested in production
├── references          # documents that help understanding the domain of the project
├── reports             # documents to communicate the project's outputs
└── src                 # data pipeline code directory
    ├── data            # data transformation scripts
    ├── models          # training scripts
    └── visualization   # scripts to communicate results
```

3.10. Dos and Don’ts

To finish this section, we provide here a short list of recommendations and pitfalls.

Dos

- **Define clear objectives:** unambiguous technical objectives, business goals and desired outcomes are together the backbone of an entire data science project.
- **Understand the domain and the business needs:** close collaboration with end users is here necessary and leads to align data science solutions with expectations.
- **Quality data collection and preparation** (GIGO: Garbage In Garbage Out): not only must raw data be of high quality to allow for high quality results, but also preparation must never lessen this quality to maintain reliability all along the project. Stay alert for missing values, outliers and biases.
- **Iterative modelling and evaluation:** doing so prevent data and model drifts, but also allows to detect issues early.
- **Reproducibility and documentation:** this is the key to collaboration, support of future work and long-term maintainability and continuity.
- **Continuous learning and adaptation:** staying updated on the latest tools and techniques, being open to implement new approaches in crucial to stay on-par with expectations around a data science project.
- **Do a pre-mortem:** take some time to identify risks and reasons why a project might fail. Think of how to avoid these.
- **Start by setting templates and tooling:** Start each project by setting up the required templates, tools and frameworks. For example, start by creating a git repository, set up the default actions that should trigger at each push (running tests for instance), each merge, etc.



Don’ts

- **Overfitting or underfitting models:** Fitting models too closely to training data (overfitting) or oversimplifying models (underfitting) can reduce predictive accuracy on new data.
- **Bias in data and models:** Data can reflect biases present in society, leading to biased models. Failing to detect and mitigate biases can result in unfair or discriminatory outcomes.
- **Relying solely on algorithms:** Algorithms are important, but understanding the problem domain and business needs as well as and the context of the data is equally crucial. Algorithms alone cannot solve every problem.
- **Data privacy and security concerns:** Mishandling sensitive or personal data can lead to privacy breaches, legal issues, or damage to an organization's reputation.
- **Overcomplicated models:** Don’t use a complex model for the sole sake of complexity.
- **Perfect is the enemy of good:** Don’t get lost into too many details, and don’t strive for perfection. Refer to the project’s goals: when they are met both in time and quality, go on with other tasks.

4. Continuous Monitoring and Improvement

Regularly assessing and refining models to ensure their relevance and accuracy over time is essential, particularly in dynamic environments. Continuous monitoring and improvement require both technical considerations as well as appropriate collaboration and communication.

4.1. Technical aspects

The technical side of monitoring and improvement is mainly focused on how to track your own progress during development and to make that the proposed models are suitable under the project requirements.

- **Data Quality Monitoring:** Regularly assess the quality of incoming data, addressing issues such as missing values, outliers, and inconsistencies to maintain data integrity.
- **Model Performance Tracking:** At the start of the project, define metrics and select data to be used to evaluate the performance of trained models. Stakeholders should brainstorm together to define a sensible evaluation strategy that is as close as possible to the intended production environment but keeps the related effort manageable.
- **Fail early:** Business requirements for data science projects can often be overwhelming at first. In these cases, it can help to perform a first proof of concept (POC) under simplified conditions to see if the solution could work at all and to better estimate the effort to model the required final product. Be aware though to not oversell a working POC with vastly reduced complexity.
- **Drift Detection and Adaptation:** Detect concept drift and distributional changes, triggering model adjustments or retraining to adapt to evolving data patterns.
- **Model Retraining and Feature Engineering Updates:** Schedule periodic model retraining using updated data and reassess feature engineering techniques to enhance interpretability and performance.
- **Algorithm Exploration and Tuning:** Stay informed about new algorithms, experiment with different options, and fine-tune hyperparameters to optimize model performance.
- **Security, Compliance, and Resource Monitoring:** Regularly review security protocols, ensure compliance with privacy regulations, and monitor resource utilization for efficient and secure data handling.

4.2. Collaboration and Communication

Effective collaboration and communication facilitate shared understanding and utilization of insights but might not directly impact the technical aspects of analysis.

- **Documentation and Knowledge Sharing:** Document model architecture, assumptions, and limitations, while fostering a culture of knowledge sharing to facilitate continuous learning within the team.
- **Feedback Loops:** Establish feedback loops for continuous improvement based on user/stakeholder input. Schedule workshops with the target users of the model or application early in the development process to ensure that the requirements detailed at the start of the project are indeed aligned with the needs of the user base. Do not hesitate to adapt requirements as needed but make sure to document and communicate these changes in a transparent and tractable way.
- **Results and performance communication:** Do not underestimate the importance and impact of great reporting and visualization. Working on a project daily, it is easy to keep a good overview of its different aspects. Decision takers who are less in touch with the development should be provided with reports and visualizations that make it easy to grasp the progress, and that are sufficiently detailed to allow for good feedback and business decisions.
- **End of project learnings:** At the end of a project, successful or failed, make it a custom to document the steps taken, challenges that occurred (especially unpredicted) and communicate the learnings within the team to enable a better knowledge transfer to other projects and allow to take precautions to circumvent experienced problems.

5. Lifecycle management

5.1. Continuous Integration (CI)

Continuous integration for data science involves automating the integration of data preprocessing, feature engineering, model training, and evaluation into a cohesive pipeline. Taking full advantage of these comprises the following practices:

- **Committing regularly** helps to quickly identify potentially breaking changes. Smaller breaking changes are easier to fix, not only because they imply less code, but also because fixing them usually occurs temporally closer to when they are introduced. In the long-term this reduces drastically the time spent in debugging.
- **The one who writes tests should not be the one who writes tested code.** Not only does this mitigates the risk of introducing identical logical errors in the code and the tests, but it also ensures code legibility and reproducibility through collaboration.
- **Write and update tests** that ensure **coverage.** This requires a deep understanding of the variety of data and use cases the project may have to deal with. Updating tests with new features as well as with new data is crucial, as both may introduce conditions that break previous assumptions implemented in tests.
- When a build or test breaks, **understand why.** It may be that the introduced change is buggy, but it may also be that the build or test is stale and must be updated to reflect introduced changes. If so, have somebody else investigating and implementing the necessary changes to the integration pipeline.
- **Include structurally representative data** and keep these structures up to date if data evolves. Doing so ensures the data pipeline can deal with live, real-world data.

5.2. Continuous Delivery (CD)

Continuous delivery for data science involves automating the deployment of models, pipelines, and associated code changes into production or staging environments. Such tasks must be lead with the goal to ensure consistency and reproducibility of data pipelines and model deployments, and involve the following practices:

- **Versioning and timestamping** of both models and data is crucial to ensure reproducibility over time: thus, a model that has been known to produce precise results on a specific dataset will be easier to reuse with the same data. Additionally, difference in models and data timestamping is essential to scrutinize to detect potential data drift or model drift.
- **Implement a rollback mechanism** that takes full advantage of the versioning and timestamping of models and data. Doing so allows for quick swaps and reversions of models and data use in deployment.
- **Environment consistency:** the use of Ansible playbooks, for instance, allows to specify the needs for the data pipeline to run smoothly, while ensuring the deployment relies on pre-known resources (processing power, memory, etc.).
- **Implement health checks** mitigates potential failures and performances degradation in deployed pipelines. This aims at verifying the functioning of said pipelines with check for input/output expectations, latency, results quality, etc.
- **Plan for collaborative deployments** helps to establish a covering of features to check when deploying a new version of a pipeline. This spans from aligning on the project's objectives, ensuring models perform as expected, services run smoothly, etc. This should enrol all members implicated in the project or its deployment, whether they are data scientists, software or IT engineers, domain experts, etc.

5.3. MLOps (Machine Learning Operations)

MLOps is a set of practices and tools that streamline the lifecycle of machine learning models, from development to deployment and monitoring. Good practices there aim at ensuring automation, reproducibility and scalability of data science workflows, spanning over model training, validation, deployment and monitoring:

- **Continuous model training** should be implemented to regularly and automatically trigger training pipelines with new data. This helps to prevent data drift and model drift by learning on new patterns and behaviours in more up-to-date data.
- **Pipeline orchestration** tools ease the management of end-to-end data analytics into data pipelines.
- Building a **feature store** aims at identifying, managing and reusing parts of inputs that must be used as inputs for any stage of a data analytics pipeline. Doing so allows to keep track of which data is mandatory for said stages to work, thus ensuring reproducibility and consistency.
- **Monitoring and logging dashboards** provide insights not only in the performance of model inferences, but also to integrate health checks of models in development and in deployment.

5.4. DevOps (Development Operations)

Similarly, DevOps is a set of practices and tools focused on aligning development tasks of a data science project with deployment and monitoring considerations. They foster collaboration and streamline processes toward reliability and efficiency of data science integration in deployment:

- **Continuous feedback loops** aim at establishing processes to orchestrate development, testing and deployments, expectedly in this order, such that every new development is tested before deployment, and information gathered from deployment drives in return new developments.
- **Incident response planning** mitigates with breaking changes by establishing repeatable procedures to deal with broken builds and tests so that they become habits over time.
- **Continuous improvement** encourages collaboration and reproducibility through the systematization of regular code reviews and refactoring sessions that strive for more efficient, more legible and more tested code.
- **Continuous security** spans the entire data science project and ensures that good security practices are always active at all stages of the project.

References

[1] Solomon Kurz, "Statistical Rethinking with brms, ggplot2, and the tidyverse", *Retrieved from osf. io/97t6w*, 2019.

[2] (2024) General Data Protection Regulation (GDPR) Compliance Guidelines. [Online]. https://gdpr.eu/

[3] "Regulation (EU) 2022/868 of the European Parliament and of the Council of 30 May 2022 on European data governance and amending Regulation (EU) 2018/1724 (Data Governance Act) (Text with EEA relevance)", *OJ L 152, 3.6.2022, p. 1–44*, 2022.

[4] "Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS", *COM/2021/206 final*, 2021.

[5] "Regulation (EU) 2023/2854 of the European Parliament and of the Council of 13 December 2023 on harmonised rules on fair access to and use of data and amending Regulation (EU) 2017/2394 and Directive (EU) 2020/1828 (Data Act)", *OJ L, 2023/2854, 22.12.2023*, 2023.

[6] (2018) Open Consultation on FAIR Data Action Plan - LIBER Europe. [Online]. https://libereurope.eu/article/fairdataconsultation/

[7] "Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts", International Organization for Standardization, Geneva, CH, Standard February 1985.

[8] Abraham Wald, "A method of estimating plane vulnerability based on damage of survivors", *Statistical Research Group, Columbia University. CRC*, vol. 432, 1943.

[9] (2024) Privacy by Design - General Data Protection Regulation (GDPR). [Online]. https://gdpr-info.eu/issues/privacy-by-design

[10] (2024) 6.4. Imputation of missing values — scikit-learn 1.4.0 documentation. [Online]. https://scikit-learn.org/stable/modules/impute.html

[11] (2024) Multivariate Imputation by Chained Equations • mice. [Online]. https://amices.org/mice/

[12] (2023) tf-idf. [Online]. https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[13] (2024) GitHub - cxli233/FriendsDontLetFriends: Friends don't let friends make certain types of data visualization - What are they and why are they bad. [Online]. https://github.com/cxli233/FriendsDontLetFriends

[14] Ruben van den Goorbergh, Maarten van Smeden, Dirk Timmerman, and Ben Van Calster, "The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression", *Journal of the American Medical Informatics Association*, vol. 29, pp. 1525–1534, 2022.

[15] Khan Md Hasib et al., "A survey of methods for managing the classification and solution of data imbalance problem", *arXiv preprint arXiv:2012.11870*, 2020.

[16] Daniel Berrar and others, Cross-Validation., 2019.

[17] (2024) Git - Documentation. [Online]. https://git-scm.com/doc

[18] (2024) Trunk-based Development | Atlassian. [Online]. https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development

[19] (2024) Git - gitignore Documentation. [Online]. https://git-scm.com/docs/gitignore

[20] (2024) Welcome - Sphinx documentation. [Online]. https://www.sphinx-doc.org/en/master/

[21] (2024) Doxygen homepage. [Online]. https://www.doxygen.nl/index.html

[22] (2024) Get started with roxygen2. [Online]. https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html

[23] (2024) GitHub: Let's build from here · GitHub. [Online]. https://github.com/

[24] (2024) The DevSecOps Platform | GitLab. [Online]. https://about.gitlab.com/

[25] (2024) Bitbucket | Git solution for teams using Jira. [Online]. https://bitbucket.org/

[26] (2024) Unit Testing for R • testthat. [Online]. https://testthat.r-lib.org/

[27] (2024) pytest: helps you write better programs — pytest documentation. [Online]. https://docs.pytest.org/en/8.0.x/

[28] (2024) GitHub - pyenv/pyenv: Simple Python version management. [Online]. https://github.com/pyenv/pyenv

[29] (2024) Introduction to renv • renv. [Online]. https://rstudio.github.io/renv/articles/renv.html

[30] (2024) Docker: Accelerated Container Application Development. [Online]. https://www.docker.com/

[31] (2024) Kubernetes. [Online]. https://kubernetes.io/

[32] (2024) Open Container Initiative - Open Container Initiative. [Online]. https://opencontainers.org/

[33] (2024) Snakemake | Snakemake 8.4.8 documentation. [Online]. https://snakemake.readthedocs.io/en/stable/

[34] (2023) A DSL for parallel and scalable computational pipelines | Nextflow. [Online]. https://www.nextflow.io/

[35] (2024) Google's R Style Guide | styleguide. [Online]. https://google.github.io/styleguide/Rguide.html

[36] (2024) Welcome | The tidyverse style guide. [Online]. https://style.tidyverse.org/

[37] (2013) PEP 8 – Style Guide for Python Code | peps.python.org. [Online]. https://peps.python.org/pep-0008/

Abreviations

- **AI** Artificial Intelligence
- **CD** Continuous Delivery
- **CGIE** Centre de gestion informatique de l'éducation (IT Management Centre)
- **CGPD** Commissariat du gouvernement à la protection des données auprès de l'État (Commissioner for Data Protection with the Luxembourg State)
- **CI** Continuous Integration
- **CNPD** Commission nationale pour la protection des données (National Commission for Data Protection)
- **CTIE** Centre des technologies de l'information de l'État (Government IT Centre)
- **CV** Cross Validation
- **DAG** Directed Acyclic Graph
- **Dev** Development
- **DPO** Data Protection Officer
- **EDA** Exploratory Data Analysis
- **FAIR** Findable Accessible Interoperable Reusable
- **GDPR** General Data Protection Regulation
- **GIGO** Garbage In Garbage Out
- **IT** Information Technology
- **JSON** JavaScript Object Notation
- **LNDS** Luxembourg National Data Service
- **ML** Machine Learning
- **OCI** Open Container Initiative
- **Ops** Operations
- **PEP** Python Enhancement Proposal
- **POC** Proof of concept
- **SIGI** Syndicat intercommunal de gestion informatique
- **SRC** Source
- **TF-IDF** Term Frequency - Inverse Document Frequency
- **(V)ENV** (Virtual) Environment
- **YAML** Yet Another Markup Language

Published by
Ministry for Digitalisation
info@digital.etat.lu
www.digitalisation.lu

May 2024