From Research to Impact: A Comprehensive MLOps Platform for AI Model Deployment in Drug Discovery

Elina Koletou [1□], Le Mu [2□], Raya Stoyanova [3], Jorge Oswaldo Gomez Muñoz [4], Krystian Kozieł [5], Paweł Cisło [5], Lorenzo Maffioli [4], Dariusz Adamczyk [4], Jakub Witkowski [4], Małgorzata Bartosik [4], Małgorzata Widelicka [4], Marius Quadflieg [1], Olga Maksymiuk-Młynarczyk [4], Santiago Gonzalez [6], Yun Yvonna Li [1], Josep Arús-Pous [1], Witold Przeciechowski [4], Łukasz Rakoczy [4], Marek Grzenkowicz [4], Monika Dyrda [4], Alin Spinu [7], Michael Reutlinger [1], Fabian Schmich [2], David Zilian [2], Lykke Pedersen [8], Daniel Butnaru [1], Angelika Fuchs [2], Anna Bauer-Mehren [2], Christophe Chabbert [3✉]

1 Roche Innovation Center Basel, Switzerland

2 Roche Innovation Center Munich, Penzberg, Germany

3 Roche Innovation Center Zurich, Switzerland

4 Roche Polska, Warsaw, Poland

5 Billennium by order of Roche Polska sp. z o.o., Warsaw, Poland

6 Roche Farma, S.A., Madrid, Spain

7 7N by order of Roche Polska sp. z o.o., Warsaw, Poland

8 Roche Innovation Center Copenhagen, Copenhagen, Denmark

□ These two authors contributed equally to this work and are joint first authors.

✉ Corresponding author

Highlights

- A novel comprehensive cloud-native MLOps platform covers the entire preclinical drug discovery process end-to-end.
- Flexible integration strategies bridge a fragmented digital ecosystem and deliver AI value to scientists for informed decisions.
- Decoupling of machine learning and operational code enhances reliability and R&D specialization.

Abstract

The increasing application of machine learning (ML) and artificial intelligence (AI) in drug discovery necessitates robust and scalable solutions for operationalizing these technologies. This paper presents the development and implementation of a cloud-native Machine Learning Operations (MLOps) platform designed to accelerate the delivery of production-ready, scalable, and reliable AI models within the pharmaceutical drug discovery environment. The platform addresses challenges associated with operationalizing AI in complex research and development settings, emphasizing the critical interplay between well-defined business requirements and robust technical capabilities. Foundational business needs, such as ensuring model trustworthiness, data traceability, and rapid experimental iteration, directly influenced the technical architecture, including choices around orchestration (Kubeflow Pipelines), real-time serving (KServe), data integration, and monitoring. Our comprehensive MLOps approach demonstrates how strategic alignment of business objectives with scalable

technical solutions can unlock significant value, showcasing practical applications across diverse ML workflow types. The platform fosters operational excellence, accelerates research in a cost-effective manner, and provides valuable insights and a proven methodology for bridging the gap between scientific innovation and large-scale AI deployment in various industries.

Keywords

MLOps, Drug Discovery, Cloud Computing, Kubeflow, Artificial Intelligence, Machine Learning

Introduction

The past few years [1–4] have witnessed a dramatic surge in the demand for machine learning (ML) and artificial intelligence (AI) models in drug discovery and early development, highlighted by transformative examples like AlphaFold [3], DINOv3 [5], MoIMIM [6], DiffDock [7] and Evo [8]. Although there are distinctions between AI and ML models, for simplicity and consistency, we will predominantly use the term AI throughout this manuscript to refer to both model types, and only explicitly mention ML when a specific distinction is required. Applications of these models range from target identification and prioritization [9,10], optimization of lead series in small and large molecule research all the way to image classification for digital pathology [11,12]. However, taking proofs of concepts in AI all the way to production has remained an important challenge for many organizations who often struggle to fully scale the usage

of their models in a sustainable manner [13–15]. Many of the challenges encountered are of operational nature and linked to the difficulties associated with deploying ML pipelines or endpoints, ensuring traceability and reproducibility of the pipelines and computational steps and finally, providing scalable solutions that are fully accessible to end users. These challenges have become even more complex in recent years with a sharp increase in the reuse and/or fine tuning of publicly available models internalized from publications or from platforms such as Hugging Face© [16,17] and NVIDIA NIM™ [18]. Failing to address and overcome these challenges can result in a slower adoption of AI in research processes and workflows, an increase of the failure rates of the AI initiatives and ultimately a loss of competitiveness [19–21]. While some teams have described successful solutions to certain aspects of drug discovery such as compound design [22], or specific challenges like multimodal applications[23], there are no known holistic approaches to operationalize models used throughout the entire discovery process.Here, we present our cloud-native Machine Learning Operations (MLOps) platform, specifically built to accelerate the delivery of production-ready, scalable and reliable AI models for drug discovery.

Results

(1) Complex Needs for AI Model Operationalization in Drug Discovery Require Flexible Solutions

As AI plays an increasingly critical role in drug discovery [24], we set out to better understand the technological needs of our research organization in these areas. To

achieve this goal, we listed key processes and workflows which either already relied on AI models or were planned to do so in the future. We listed these processes & workflows together with the models that support them: we identified 28 use cases requiring partial or full operationalization of AI. Of note, the majority of such processes (n=17, 60.7%, Table S1) needed several models to fully leverage the potential of the technology and deliver improvements in the drug discovery value chain. The expected impact of these use cases spanned the whole drug discovery and development pipeline, from preclinical phases all the way to Phase 3 clinical trials. However, as the majority of the proposals (n=18, 64.3%, Table S2) originated from preclinical phases (i.e. before entry into humans), we focused our efforts on providing solutions fit for purpose in preclinical and non-validated environments.

We analyzed these use cases by systematically collecting information on technical requirements for training, inference and validation workflows, as well as the required input data for each of the pipelines. We also conducted multiple interviews with scientists developing models together with their users and other important systems and applications owners in our organization. Key requirements and needs were extracted and formalized into user stories and ranked based on their importance and urgency (see method section for a more detailed overview of our methodology).

The first important finding from this analysis is that business processes relying on the output of models had very different serving needs; from online, real-time serving, sometimes with very low latency to accommodate demanding user experience, all the

way to complex parallelisation of workflows to simultaneously process large amounts of images or objects, sometimes with speed. We even had processes requiring fast and efficient out-scaling of the delivery of model outputs in the context of surges of traffic and demand for predictions. Although the majority of models (n=19, 67.9%, Table S3) were expected to impact less than 50 end users, they were impacting critical business processes and based on their diverse serving needs we were able to get some first estimates of the type of scalability requirements for our platform.

A lot of the use cases considered highlighted the importance of demanding computational needs such as GPU access (n=11, 39.3%, Table S4). As establishing reproducible training pipelines is crucial for ensuring transparency and traceability in the model development process, we considered this as a key requirement for our platform. This need for GPU access later extended to inference pipelines due to the rapid development of use cases leveraging transformers technologies, such as embeddings generation [25–28].

Interestingly, while frequent and automated re-training is considered best practice in ML, only two thirds of the surveyed use cases have considered these needs (n=18, 64.3%, Table S5) while for one third of the cases the frequency of re-training was expected to be very low (n=8, 28.6%, Table S5), often due to the difficulty to obtain new relevant training datasets, or was not considered at all as a requirement (n=2, 7.1%, Table S5). We therefore concluded that not every use case would need to automatically trigger training pipelines and that manual control of the training workflows was also necessary

in many cases.

For all cases, the need for a reliable, scalable and functional integration with other systems was very prominent. This was also a key requirement identified by our stakeholders which we chose to prioritize. This requirement is particularly critical and challenging as our use cases rely on a wide variety of data modalities, including structured tabular data (n=14, 50%, Table S6) and unstructured data like text (n=5, 17.9%, Table S6) or images (n=5, 17.9%, Table S6) to deliver their output.

As our models are processing valuable data assets subject to very high levels of confidentiality, we also concluded that any technical solution built to operationalize AI in our organization should adhere to the strictest security, data protection, and compliance standards required, such as GDPR(General Data Protection Regulation) [29] and EU AI act [30].

Finally, our analysis revealed that while most models are developed using Python as the main programming language (n=15, 53.6%, Table S7), some rely on the statistical language R (n=9, 32.1%, Table S7) and sometimes on proprietary languages such as Matlab. We therefore concluded that any solution that supports operationalization of models should support at least Python and R as programming languages used to develop models. The aim behind this effort was to eventually be able to reach higher MLOps maturity levels that consider the tools and platforms to be completely language agnostic [31,32].

These findings guided the development of a computational platform to efficiently support the wide variety of the needs for the operationalization of AI in drug discovery.

(2) Building the MLOps Platform: A Scalable and Modular Infrastructure Leveraging Kubeflow and KServe for Three Primary ML WorkflowTypes

We listed the requirements emerging from our key processes and translated them into concrete technical requirements to steer the creation of an MLOps platform to operationalize AI models. In particular, we identified that the diverse computation needs could be met by providing an infrastructure able to cope with **3 primary types of ML workflows**:

1. **(Re-)training workflows:** The jobs supporting these workflows tend to run over extended periods, but their traffic pattern is periodic - most of these jobs typically check for new data availability at given frequencies to ensure models are only trained when necessary.
2. **Real-time (or online) inference workflows:** The jobs supporting online inference workflows usually complete within a shorter time frame, but their traffic pattern is characterized by unpredictable spikes. Low latency and high throughput to deliver inference is a key requirement for these jobs.
3. **Batch (or offline) inference workflows:** The workflows are commonly used for data modalities that require significant computational resources, such as large images, and where real-time inference is not necessary or possible. Batch inference jobs supporting such workflows often run for longer times, and their

traffic pattern is sporadic, depending on the need for inference generation.

As all of these jobs can be computationally and/or memory-intensive, we reasoned that the computational infrastructure of our MLOps platform would require both horizontal (adding more machines or nodes) and vertical (increasing the power of existing machines, such as adding more CPU, RAM, or storage) scalability to accommodate these needs.

Our current on-premises High Performance Computing (HPC) infrastructure failed to satisfy these requirements due to a lack of elasticity as well as reduced availability and scalability (both horizontal and vertical). In particular, we observed that many users faced growing challenges when running online or batch inference jobs and workflows. These limitations prompted us to adopt a cloud-first approach for our MLOps platform, and to therefore opt for cloud-native technologies. Indeed, cloud infrastructures offer on-demand access to computational and storage resources in a modular manner. We reasoned that it would allow our users to dynamically scale horizontally and vertically based on the nature of their ML workflows, thereby fulfilling the requirements described above.

Table 1: Key Components of our Scalable and Modular MLOps platform. This table summarizes the key technologies utilized in the scalable and modular computational infrastructure of our MLOps platform for handling ML workflows, detailing the primary function of each component and the criteria that necessitated its selection. These components are articulated in the broader architecture shown in Figure 1.

Key Functionality	Technology Selected	Key Criteria
MLOps Platform Foundation & Container Orchestration	**Amazon Web Services (AWS) EKS / Kubernetes**	Provides the necessary **horizontal and vertical scalability** for diverse, computationally intensive ML workflows. Adoption of a **cloud-first approach** resolved limitations of previous on-premises HPC infrastructure.
ML Pipeline Orchestration	**Kubeflow Pipelines**	Selected as the central orchestrator for reliable, scalable training and batch inference workflows. Chosen for native **Kubernetes integration** and flexibility for custom DAG-based workflows.
Real-time Model Serving	**KServe**	Enables the serving of real-time (online) inference jobs with **low-latency** and **auto-scaling capabilities**.
API Management & Gateway	**Gravitee.io**	Manages REST APIs, ensuring **standardized access**, provides **granular Role-Based Access Control (RBAC)**, and allows for monitoring of API health.

Temporary Data Storage	**AWS Elastic File System (EFS), AWS Elastic Block Store (EBS), and S3 buckets**	Provides temporary locations to host data (e.g., features, model binaries) **close to computation** to minimize data transfer costs and time. These accounts are managed independently for resilience.
Observability & Performance Monitoring	**Datadog, Grafana Loki, Grafana Dashboards**	Metrics and logs are collected and monitored to ensure **adequate observability**, enabling the MLOps platform team to track performance bottlenecks and model usage patterns.

To build our MLOps platform, we leveraged Amazon Web Services (AWS) cloud services. We used Infrastructure as Code (IaC) to effectively manage and provision such infrastructure resources. IaC allowed us to create, modify, and remove infrastructure resources in a consistent and repeatable manner, reducing errors and improving efficiency. It also promoted collaboration among team members and ensured that infrastructure changes are well-documented and versioned, thereby facilitating traceability and auditability.

Building upon this foundation of robust infrastructure management [33,34], we then focused on the specific requirements of our ML workflows. As (re-)training jobs and batch inference jobs require robust, scalable, and flexible pipelines, we selected

Kubeflow as a central ML pipeline orchestrator for our MLOps platform. By deploying Kubeflow on AWS EKS, we therefore were able to leverage Kubernetes' inherent scalability, enabling efficient resource management and scalability of ML workflows in response to demand. Indeed, Kubeflow Pipelines facilitate the definition of ML workflows as Directed Acyclic Graphs (DAGs), with each step encapsulated in a Docker container. This containerized approach provides automation capabilities that minimize the risk of human error and increase reproducibility. It also ensures isolation, reproducibility, and the ability to execute each step with specific resource requirements. By tailoring resource allocation to the needs of each pipeline step, we optimize computational resources. For instance, simple tasks like data schema validation, which have lower memory needs and computational requirements, can leverage cost-effective CPU instances. In contrast, resource-intensive training jobs, which rely on frameworks like TensorFlow or PyTorch and require substantial memory benefit from the parallel but costly processing power of GPU instances. Finally, thanks to Kubernetes' dynamic resource management and serverless scalability principles, resources are adjusted based on demand, further enhancing cost efficiency and ensuring that training and batch inference workflows seamlessly handle varying levels of load without manual intervention.

Alongside Kubeflow, we deployed KServe on AWS EKS to run real-time inference jobs. KServe is a robust AI model serving platform designed to deploy and manage AI models at scale, with features like automatic scaling and seamless platform integration also utilized by companies such as NVIDIA[35], RedHat[36], Ubuntu[37],

Bloomberg[38], Cisco[39], and Hewlett Packard Enterprise[40]. KServe's standardized and scalable approach to serving models helped us achieve low-latency and high-throughput, which is critical for real-time inference jobs. Additionally, KServe's auto-scaling capabilities ensure that our InferenceServices can dynamically adjust to varying loads, scaling down to zero when not in use, thereby optimizing resource utilization and cost.

As all 3 types of ML workflows require access to data, we had to consider integration with data sources as a critical component of our MLOps platform. Due to the very complex data landscape in drug discovery (see results section 3), we did not intend to maintain primary data storage on our MLOps platform, but rather provided temporary storage locations backed by AWS Elastic File System (EFS), AWS Elastic Block Store (EBS), and S3 buckets. These temporary storage locations are intended to host data such as pre-calculated features and large model binaries, keeping the data close to the computation. This approach helped minimize the traffic required to move data from on-premise infrastructure to the cloud and vice versa, and therefore reduced costs and computation time.

Moreover, to ensure adequate observability of our entire solutions and of the models deployed, we invested in proper monitoring service tools. We relied on Datadog to collect metrics describing usage of all of our models in production, including the number of predictions or output requested over time either via API or inference pipelines. The ability to observe which clients requested such outputs helped our scientists refine their

understanding of model usage and therefore of their impact. We also collected logs to get richer context such around usage (such as details about the payloads from inference requests) and improve our understanding of usage patterns. Logs were collected using Grafana Loki[45] and accessed through Grafana Dashboards[46].

Ultimately, the combination of Infrastructure as Code, Kubeflow Pipelines, and KServe has created a comprehensive, end-to-end solution for deploying and managing AI models within our drug discovery workflows. This strategic decision effectively addresses the limitations of our previous on-premises HPC infrastructure. In addition, it ensures the scalability and cost-effectiveness of our operations and also fosters operational excellence by automating complex tasks, optimizing resource allocation, and providing a robust foundation for handling diverse workflow patterns.

The overview of the architecture of our MLOps platform's infrastructure in Figure 1 shows how different modular components are orchestrated and integrated to meet our computational needs. The container orchestration system (Kubernetes) is the foundational infrastructure service which supports the deployment, scaling, and management of all the ML containerized workflows that are running on our MLOps platform. The ML pipeline orchestrator (Kubeflow) and API services (relying on KServe for autoscaling) provide key functionalities as they allow users to train models and deploy them in production, either via pipelines or REST API endpoints. They provide the abstraction required for the successful containerization and deployment of each ML

workflow which runs on the container orchestration system. Once trained, models are stored in a registry, together with relevant metadata. Cloud-based data storage accounts, data streaming tools and APIs facilitate integration with the rest of the digital ecosystem and are managed independently of our container orchestration system to improve the resilience of our MLOps platform. In this context, by digital ecosystem we refer to the ensemble of other systems, data products, and user-facing applications with which our MLOps platform integrates but are not part of it. More details about the technical solutions used for those integrations can be found in the next section and on Figures 2.1-2.4. Of note, the different components of the digital ecosystem may be deployed on premise or on other cloud landing zones. Finally, the API management platform and the Monitoring Service tools ensure adequate observability of our entire solutions and of the models deployed, both for our MLOps platform team and for the scientists deploying models.

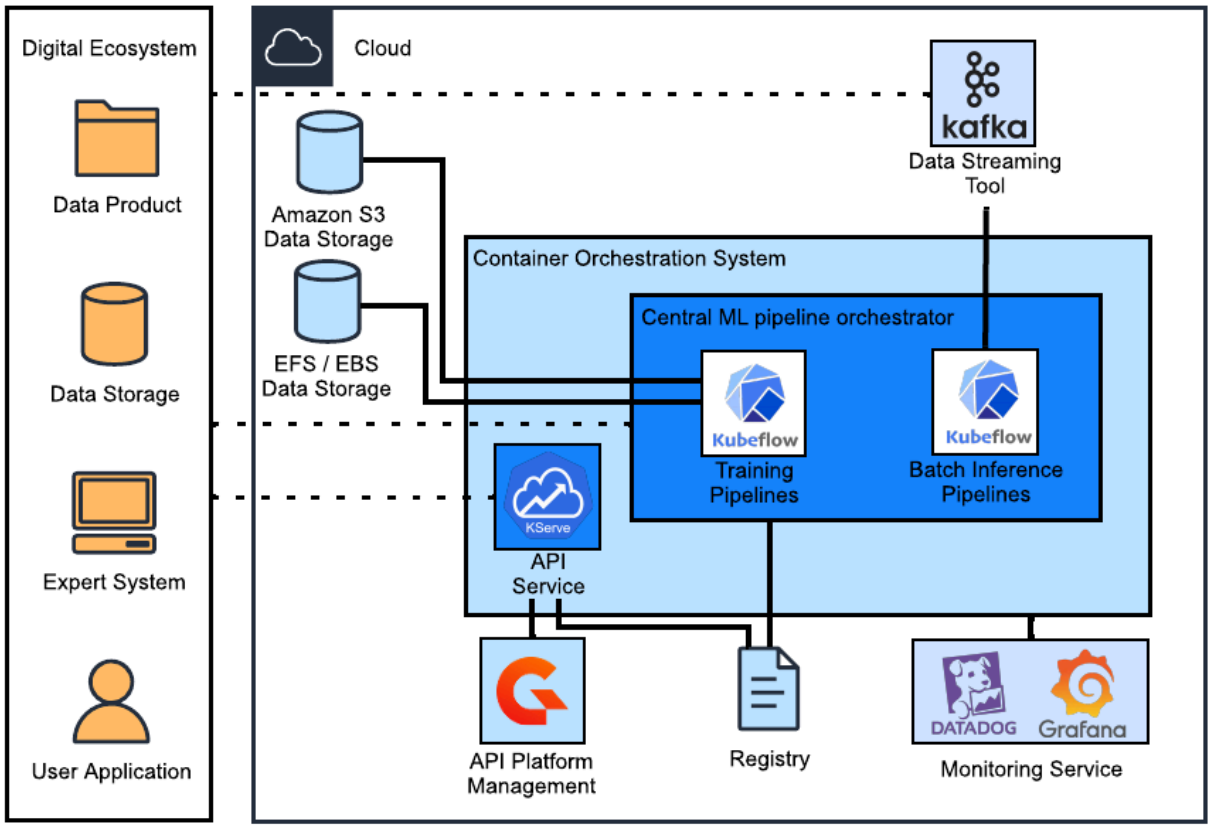

Figure 1. **The architecture of our MLOps platform.** The key components of our MLOps platform architecture are deployed using cloud native technologies and depicted on the right end side of the figure. Integrations between the components of the MLOps platform are highlighted in continuous lines. Integrations between the MLOps platform components and the rest of the digital ecosystem are represented in dashed lines.

(3) Flexible System Integration Strategies for Functional Data Pipelines and AI Model Output Delivery

The analysis of our use cases highlighted a significant challenge: the sheer number of our digital ecosystem components required to host critical input data for our ML pipelines almost mirrored the number of use cases themselves, pointing to a highly

fragmented digital landscape. This analysis also showed that the scientists that are relying on predictions and other AI model outputs, they also heavily use the digital ecosystem to interpret and leverage these outputs. This trend, also corroborated in a recent study [41], results in a very high number of integration points between those different components of the digital ecosystem used in the research organization.

To successfully integrate our MLOps platform in this digital ecosystem and meet user requirements, we therefore resorted to multiple system integration strategies (Figures 2.1-2.4):

1. **Kubeflow components: a direct system integration from Kubefow pipelines.**

   As each component (or step) in a Kubeflow Pipeline may be used as an execution runtime for Python code, we leveraged this capability to build a collection of connectors to key parts of the digital ecosystem. These connectors were built using Python code packed into Docker images which are executed from the Kubeflow pipeline tasks (more details may be found in the following section). We used such connectors to ingest training data, model binaries but also to publish inference and other model outputs into relevant components of the digital ecosystem. The use of versioned Python code to create these connectors made them highly reusable across pipelines and models requiring similar systems integration. They could also be customized whenever necessary to meet specific technical and/or functional requirements. In addition, the unified, versioned code

base made maintenance of the interfaces leaner and faster, thereby contributing to the scalability and reliability of the MLOps platform.

Finally, such components could also be shared within the community of engineers and scientists actively building pipelines, which encouraged collaboration and exchange. In total, we built just a handful of connectors to deploy hundreds of unique Kubeflow pipelines in production. They allowed us to ingest data from compound registration systems, data warehouses critical for medicinal chemists but also data lakes hosting microscopy images. Many components were used both in training and batch inference pipelines, highlighting the benefits and portability of the approach.

2. **Streaming services: asynchronous and flexible integrations between our MLOps platform and the digital ecosystem.**

   A few of our use cases required the rapid generation of inference following specific events (such as new compound registration in a database) or even very large batch processing over all the records of a data warehouse. This was the case for many compound property predictions that needed to be generated for entire libraries of large or small molecules. The use of traditional approaches to satisfy these requirements often proved unsuccessful. Indeed, relying mostly on API calls, cron-job or manual triggering of pipelines proved either too slow or unreliable due

to the complex digital ecosystem and often tight coupling of its components.

To overcome these challenges, we used Confluent Kafka as an event streaming and queueing service to maintain a loosely coupled, and sometimes asynchronous integration between our MLOps platform and key components of the digital ecosystem.

The publication of messages describing events of interest into relevant Kafka topics helped us generate all required inferences while tightly controlling the computational load on our MLOps platform. We also used specific topics to transfer these inference results to the appropriate consumers from pipelines or microservices (see point 4 below), thereby ensuring timely and accurate delivery of model outputs.

In addition, by defining and enforcing schemas for the messages exchanged between our MLOps platform and the digital ecosystem, Kafka helped us maintain data integrity and reduce errors caused by inconsistent data formats. Finally, Kafka's fault-tolerant architecture and replication mechanisms helped us to ensure reliable delivery of model outputs, even in the event of hardware failures or network disruptions. This increase in reliability was particularly important to foster trust in the technical solutions delivering model predictions.

3. **API gateways: Gravitee.io and standardization of API access.**

Many of our use cases required an efficient approach to serve online predictions. This was generally achieved thanks to the creation of scalable inference endpoints, supported by Kserve (see section above). However, as our MLOps platform usage grew, both the number of endpoints and the number of requests per units of time also substantially increased. To cope with the scalability challenged, we introduced the API gateway Gravitee.io to effectively manage our REST APIs. The introduction of this gateway provided 5 key benefits to our MLOps platform and users:

- **Standardized API Access:** Gravitee.io allows us to define and enforce a consistent definition for all our REST APIs. This standardization simplifies the process of integrating with our APIs and ensures a seamless experience for consumers.
- **Granular Access Control:** We leverage Gravitee.io's authentication and Role-Based Access Control (RBAC) mechanisms to manage access to our APIs. This enables us to control who can access specific APIs, ensuring the security and privacy of our data.
- **Comprehensive API Analytics:** Gravitee.io provides us with detailed analytics and reporting capabilities. We can monitor API usage patterns, track performance metrics, and measure the impact of our APIs on drug discovery programs. This data-driven

insights help us make informed decisions and continuously improve our APIs. We used Datadog, Grafana Loki and Grafana Dashboards for collecting and monitoring metrics and logs that inform model owners and our MLOps platform team on model usage. These were also used to assess the impact of the MLOps platform on drug discovery, as detailed in the result section 5.

- **API discoverability:** The rich metadata associated with each API, together with granular access control helped us safely increase the discoverability of existing model endpoints throughout the organization. This can be particularly impactful in the context of research to avoid duplication of the efforts required to build models requiring very deep and domain-specific expertise.
- **API health and performance monitoring:** Gravitee.io built-in capabilities to monitor APIs health and performance helped us increase the reliability of our MLOps platform and quickly respond to incidents affecting users.

4. **Microservices: Ad-hoc integrations between our MLOps platform and the digital ecosystem**.

   Despite the very high flexibility of the 2 approaches mentioned above, we could not always successfully rely on them to integrate our MLOps platform with systems built using legacy software engineering paradigms. These particular cases therefore required dedicated microservices that were built with a single

use in mind. While we only relied on this approach in a handful of cases, it proved to be instrumental for the success of our MLOps platform in delivering AI model predictions to scientists in the right context and throughout the digital ecosystem.

We used the strategies described above to integrate our MLOps platform with relevant components of our digital ecosystem. However, through our requirement analysis, we also identified multiple data analysis workflows requiring filtering data records using ML-generated value as criteria. Indeed, we found this to be a very common need for protein engineering as well as medicinal chemist teams that need to prioritize compounds to proceed through the various stages of drug discovery projects such as lead identification and optimization. To address this challenge, we chose to permanently store the relevant AI model predictions required for these queries and workflows. Indeed, once stored in an appropriate data management system (usually a relational database), these predictions could be consumed by downstream applications and data warehouses to constrain searches and data queries. We exclusively relied on existing data management solutions to store such predictions and ensure that they would be readily and easily accessible to scientists who need them for their daily work. We then integrated such inference stores with our MLOps platform using the streaming services or the Kubeflow components described above (integration strategy 1 and 2).

Across all our use cases, we experienced that integration with the digital ecosystem is crucial to deliver model outputs in the right contexts, in other words delivering model

outputs where and when the scientists need them. Using the panel of integration strategies described here, we were able to always adapt to the technical constraints of our digital ecosystem and to deliver inference at scale, across multiple stages of the drug discovery process. This was however only possible thanks to reliable workflows and a good management of the operational aspects of AI models which we will describe in the next sections.

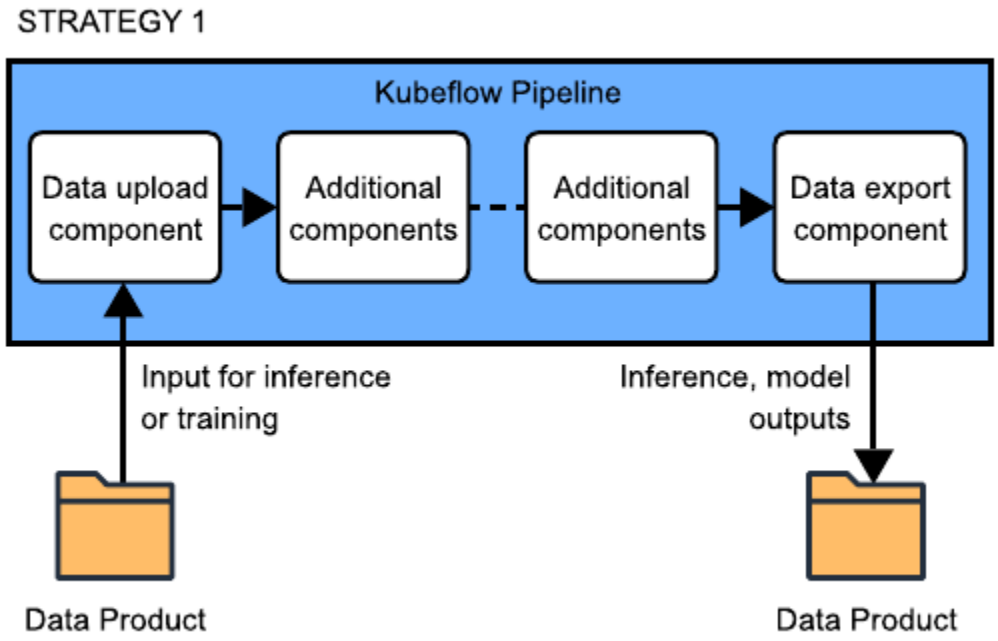

Figure 2.1 **Integration strategy 1: Kubeflow Components**. Re-useable Kubeflow components are deployed to upload and export data during the ML pipeline runs. These components provide a direct integration between the digital ecosystem and our MLOps platform for training and inference pipelines alike.

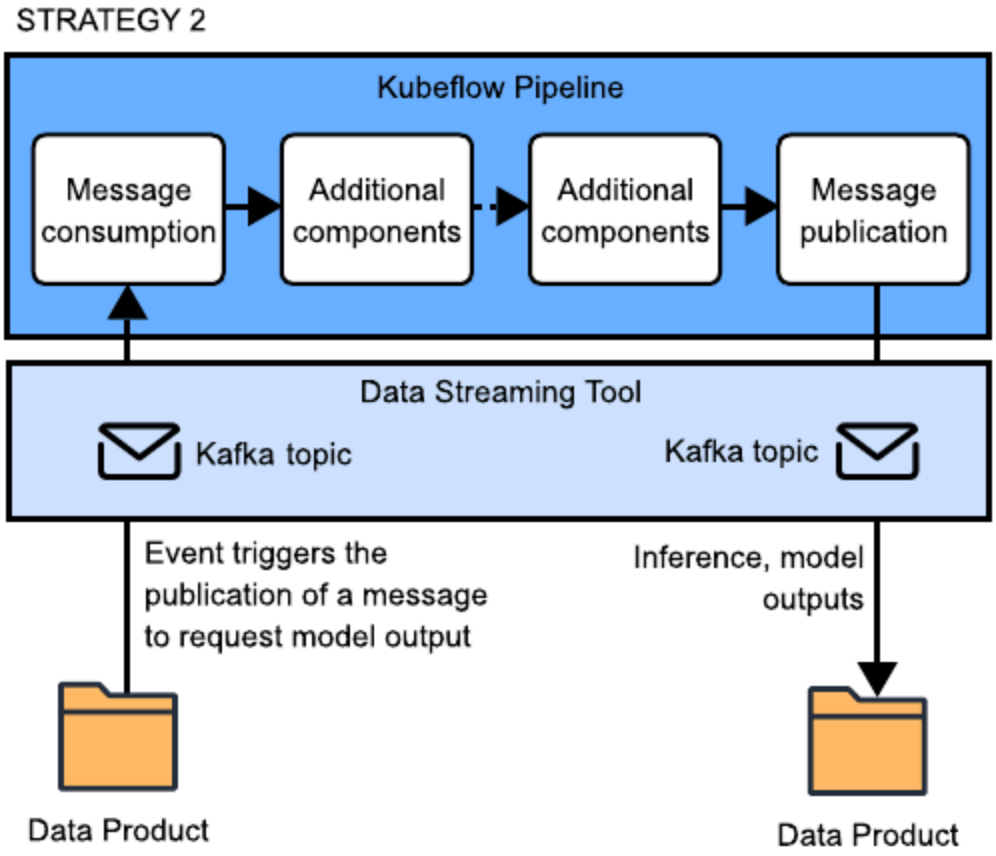

Figure 2.2 **Integration strategy 2: Streaming Services**. Data streaming services provide a loose and asynchronous integration between our MLOps platform and other components of the digital ecosystem. The use of messages and queues helps control the computational load on our MLOps platform while the enforcement of consistent schemas on data models provides consistency across the entire landscape.

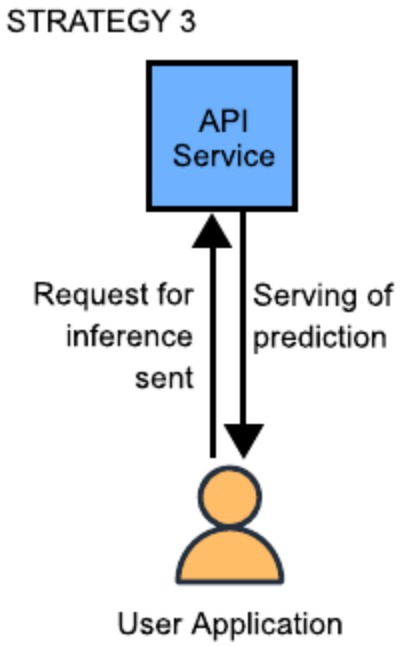

Figure 2.3 **Integration strategy 3: API gateways**. Inference data may also be served directly via REST API endpoints. These endpoints are managed and catalogued using Gravitee.io, the API management system.

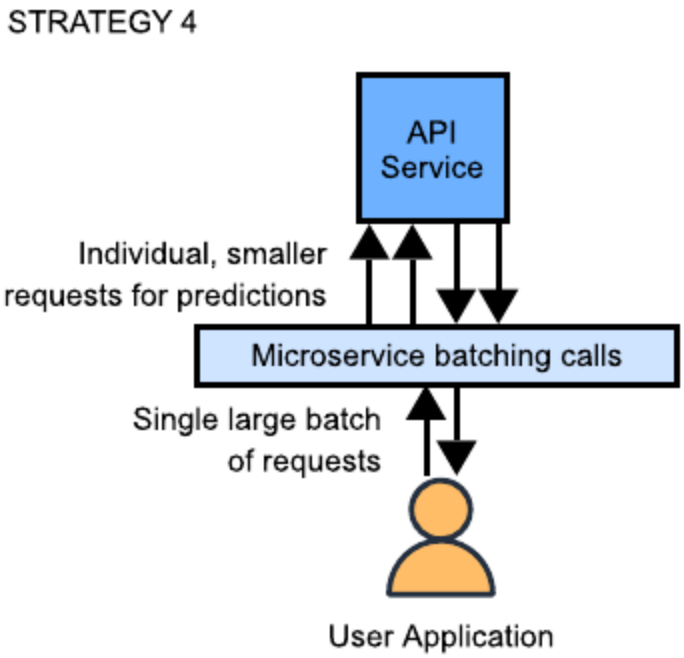

Figure 2.4 **Integration strategy 4: Microservices**. Microservices are sometimes used to build ad-hoc integrations requiring complex workflow management. In this case, a microservice batches calls and distributes them across various inference endpoints. The returned predictions are batched together before being published as a single table to the relevant components of the digital ecosystem, for consumption by scientists in the organization.

(4) Decoupling ML and Operational Code for Artifact Management: Enhancing Specialization, Focus, and Reliability of Critical Workflows

Managing key artifacts effectively and with traceability is crucial for ensuring the successful deployment and operation of AI models. In order to provide a higher level of autonomy and flexibility to scientists developing models, we opted for a separation of concerns between the ML code and the operational code as shown in Figure 3. This

separation involves decoupling the people and processes responsible for AI model design from those responsible for model deployment and operations. We implemented this separation by systematically creating 2 separated code repositories (see the Data Availability section for the urls) serving distinct purposes: one of these repositories contains the ML code itself and is primarily managed by the scientists developing the model while the second repository contains all the code necessary for the successful operationalization of the model.

The operational code encompasses the necessary components for the relevant Kubeflow pipelines, which are defined as “portable and scalable definition[s] of a ML workflow” [42] which in our case usually are training and inference pipelines. These pipelines are designed to handle tasks such as model training, validation, and deployment. The code defines the components within these pipelines, ensuring that each step is properly configured and executed. The CI/CD pipeline in the operationalization repository automates the integration, testing, and deployment stages, ensuring that any changes to the codebase are seamlessly deployed to Kubeflow, our pipeline orchestrator. The actual code executed in each component (or step) of the Kubeflow pipelines actually originates from the code repository which is primarily maintained by scientists creating the AI model. This code repository is containerized using Docker, via a Dockerfile that specifies the base image, dependencies, and the environment setup. The resulting container includes all the relevant code and environments needed for the pipelines and is stored in a Gitlab container registry, which

the operational code can access. Access to the container registry is managed to ensure security and compliance with industry standards.

We observed several benefits to decoupling the AI model design and deployment artifacts via our solution. The first key benefit was a strong gain in specialization and focus as scientists designing AI models can concentrate on developing and refining models. This is particularly critical in drug discovery as only data scientists with domain expertise on biology, chemistry and specific disease areas can ensure that AI models remain purposeful and will impact the value chain. Conversely, the deployment and operations team can focus on ensuring that models are deployed and operated efficiently and reliably. This way, DevOps best practices are not sacrificed at any point in the model life cycle and operational excellence is maintained. This resulted in an increased reliability of the ML workflows when compared to deployments managed by scientists themselves. We also observed a significant drop in incidents without hindering new development on the models deployed in production. A second benefit to this separation of concerns is the increase in productivity and operational efficiency. By focusing on the deployment of multiple pipelines and endpoints, engineers working on operationalization are more likely to identify patterns and pinpoint reusable components in ML pipelines but also in the rest of the technological stack. This often resulted in a faster automation of key pipeline and computational steps thus reducing the operational footprint of deploying multiple models in production. Finally, this separation of concerns also helped us navigate the proliferation of tools and platforms dedicated to model development. Indeed, by focusing on an integration at the code repository level, we

could abstract the operationalization of models from the initial, experimental development. This also provided more freedom to scientists in drug discovery who are constantly experimenting with new methods and approaches to AI.

The separation of the operational code also brought some challenges. The first and main one is that, prior to a productive release, the ML code must be refactored and adjusted to integrate seamlessly into the operational pipeline(s). Difficulties may arise due to scientists' lack of expertise in writing operational code and the MLOps team's unfamiliarity with the scientific context of the ML code and model. We addressed this challenge by establishing a predefined pipeline structure in advance, enabling scientists to refactor their code accordingly while retaining its ownership. This pipeline structure enables both scientists and the MLOps team to determine the appropriate level of code modularity based on various factors, such as the resource requirements of different pipeline components. By adopting this approach, scientists are responsible for creating modular code, while the MLOps team can maintain the pipeline and more accurately identify potential points of failure within it.

Another way that has helped overcome the operational code challenges was by fostering a culture of writing well-structured code and educating scientists to consider its future operational use. This aspect was reinforced with the introduction of checks and guardrails, such as standardized pre-commit hooks that automatically fix issues related to code formatting, style conventions, static typing, and security vulnerabilities. The release of standardized project templates and reusable CI/CD components, also ensure

a consistent, pre-defined structure for all AI projects and streamline their integration into operational pipelines. Indeed, the introduction of such reusable and curated components not only helped speed up development but also contributed to increase the security on our MLOps platform as they were kept continuously up-to-date to ensure minimal common vulnerabilities and exposures. Finally, these approaches were made even more successful by the advance of generative-AI powered coding assistants which greatly facilitate and accelerate code re-factoring.

Another challenge pertains to the lifecycle management of AI models. Once an operational pipeline is established, new model versions or code changes may be introduced over time. Without proper guardrails and tests, these changes can trigger the need for significant adjustments to the operational code but also incidents and workflow failures. Consequently, we established several strategies to mitigate this risk throughout the lifecycle of models in production. The first approach consists in systematically incorporating automatic testing within CI/CD pipelines to ensure that breaking changes are identified and corrected prior to deployment. Another, complementary measure is employing robust version control for both code and models, which tracks changes and allows for easy restoration of previous versions if needed. Continuous monitoring of model performance is also essential, as it enables early detection of issues, facilitating timely interventions and adjustments (see section below). Lastly, conducting thorough reviews of any code changes is crucial for effective long-term model management.

While technology is helpful in overcoming some of the challenges, communication channels and regular interactions remain critical to prevent the emergence of silos and reduce the complexity still inherent to releasing AI models in production.

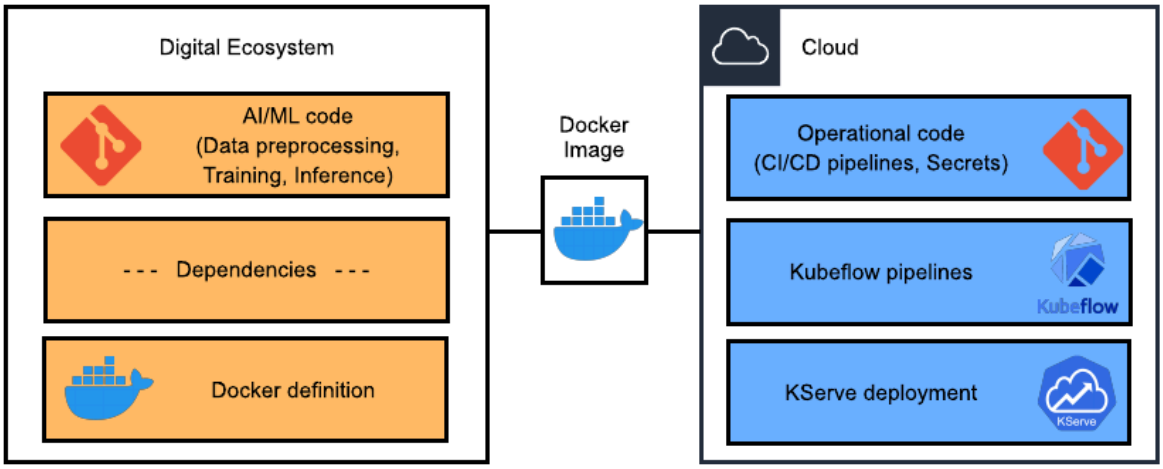

Figure 3. **Managing Artifacts.** The figure shows the separation of ML and operational code. It illustrates the decoupling of ML code (left) from Operational Code (right) that orchestrates it via Kubeflow Pipelines and deploys model endpoints using KServe. The logos next to each component are highlighting the associated technologies: GitLab for managing both code management, Kubeflow for orchestration, KServe for model endpoint deployment and Docker for containerization.

(5) Enabling Observability and Interpretability to Measure Business Impact: Collecting Usage Metrics and Contextual Artifacts for Production Models

Bringing AI models to production still represents a significant effort, in particular when their integration in relevant tools and processes is required. Over the past six years, we observed that it took sometimes weeks to months for our team to successfully deploy

well integrated predictors and pipelines in production for our 28 use cases which represent more than 150 models altogether.

In the context of drug discovery, this high investment is only justified by the anticipated business value and impact that these models will yield over time. We considered a widely cited pharmaceutical value equation[43] to identify the relevant metrics that could be impacted by the operationalization of models. Indeed, we anticipated that using MLOps frameworks and tools in drug discovery would have the following impact:

- An increase in the value of in-silico predictive power over time by combining model serving and model (re-)training in an iterative and coherent active-learning-like loop, supporting novel approaches like lab-in-the-loop in drug discovery[44].
- An increased number of AI models that are being deployed to business by increasing the efficiency of many processes in drug discovery and therefore allowing for more projects to run in parallel.
- An increase in the probability of technical success thanks to a greater likelihood of successful transition from one stage to the next within the drug development lifecycle. This can be achieved via a widespread use of models that help scientists design drugs with superior properties.
- A decrease in cycle time thanks to the automation of repetitive and time consuming tasks such as image annotation and segmentation for example.

- A decrease in costs thanks to accelerated processes and a reduction of the personnel time required to progress drug projects through the development cycle.

In practice, we observed that quantifying the impact of AI models in productions on these key business metrics is challenging. Indeed, as drug discovery is a highly technical, multi-step process with continuous improvements stemming from technologies other than AI, it is very difficult to quantify the exact impact of a model released in production.

To overcome this challenge, we reasoned that an MLOps platform could provide some technical capabilities allowing scientists and other researchers to measure metrics that could be indicative of impact and value. Therefore, we invested in establishing monitoring capabilities for the models put in production, so that we could effectively capture rich data about usage patterns. This allowed us to better understand in which context models are used, how often and to sometimes automatically list drug projects relying on the predictions for decision making. This data is also critical to ensure that our MLOps platform is able to cope with the fluctuating number of requests to serve outputs over time (see previous sections for more details on the architecture of our MLOps platform).

We also engineered reusable components and systems integration to enable scientists to export key metrics and artifacts that make monitoring of the model performance possible. These include key cross validation metrics, specifically generated predictions

or even intermediate image masks and saliency maps created during image segmentation and classification. Making such artifacts available to end users and scientists also increased the interpretability and explainability of the models deployed on the MLOps platform, which is crucial to gain the trust of domain experts and decision-makers.

In the case of compound property predictors, we also observed that end-users of the models often request access to uncertainty values associated with AI model predictions of small molecule pharmaco-kinetic parameters. This information helps them assess the reliability and confidence of the model's outputs, which is crucial for making informed decisions. We provided the appropriate computational infrastructure within our MLOps platform and the necessary systems integration in order to return these values in those systems of the digital ecosystem that end users rely on in their daily business. This results in stronger adoptions of the model in key decision making steps.

Overall, our work on productizing several use cases showed that, while it remains challenging to measure the value of AI models to production, implementing solutions to help scientists export key metrics and artifacts that monitor usage or help explain and interpret models is critical for adoption and impact tracking.

Methods

The methodology implemented for this study was built upon an iterative cycle of requirements analysis, cloud-native platform selection, and the engineering of

operational strategies. To understand the operational needs for deploying AI models in drug discovery, a rigorous requirements gathering and analysis phase was conducted, which included interviews with data scientists across many groups and hundreds of unique user stories. We conducted further business analysis by systematically collecting information and statistics from 28 model proposals. We listed the type of data modalities required for training and inference (e.g., tabular, text, images, molecules), the programming language used to develop the model (e.g., Python and R), the impacted stages of the drug discovery process (e.g. Lead Identification, Lead Optimization), information about the anticipated computational needs of our MLOps platform (e.g. GPUs, CPUs) and finally the different components of the digital ecosystem hosting critical datasets for the models and its users. This systematic process demonstrated that model deployment requires support for multiple data modalities, multiple programming languages, and different serving strategies to accommodate diverse business needs. The MLOps platform was designed to achieve a Level 2 MLOps maturity, the highest characterized level according to the widely referenced Google MLOps framework[45], focusing on fully automated ML pipeline creation and deployment. The key MLOps services in scope included model training and re-training, real-time and batch inference serving, artifact management, and comprehensive observability (monitoring and logging), while excluding manual experimentation (supported by additional third party tools) and in some cases continuous training where it was not needed. Orchestrated experimentation was made possible thanks to the full integration and automation of all components.

The derived technical requirements necessitated the design of an MLOps platform with scalable and modular computational infrastructure. Due to identified limitations in elasticity and availability within existing on-premises High Performance Computing (HPC) infrastructure, a cloud-first approach utilizing Amazon Web Services (AWS) was adopted. Infrastructure as Code (IaC) using Terraform the AWS Cloud Development Kit, and eksctl, the Amazon EKS command-line utility tool, was employed for consistent resource provisioning and auditability. The MLOps platform architecture was defined to support the three primary workflow types: scalable training, batch inference and real-time inference. Specifically, Kubeflow Pipelines deployed on AWS Elastic Kubernetes Service (AWS EKS) served as the central orchestrator for scalable training and batch inference pipelines, while KServe (on AWS EKS) was utilized for low-latency, real-time inference jobs. Overall, Kubeflow was chosen for its strong community support, native Kubernetes integration, and flexibility for custom pipeline components, features not equally supported by alternative orchestrators. Specifically, Kubeflow Pipelines was chosen as the central orchestrator due to its native support for complex DAG-based ML workflows and its tight integration with Kubernetes for scalable resource allocation. Similarly, KServe was selected for real-time serving for its low-latency, auto-scaling capabilities, critical for demanding user experience.

Furthermore, to ensure functional data pipelines and contextual output interpretability (e.g., for predictions used in the digital ecosystem), a strategy for flexible system integration was adopted, including building reusable Kubeflow components for direct system integrations, utilizing event streaming (Confluent Kafka) for asynchronous

transfers, and implementing the Gravitee.io API gateway for centralized management[46], access control, and monitoring of online prediction endpoints. Kubeflow components were created using an internal Python library wrapping the Kubeflow SDK. All code was versioned on Gitlab and all endpoints used for inference serving were registered in the API gateway for findability and better access control.

Finally, operational experience dictated specific strategies for artifact management and impact assessment. To increase the reliability of critical workflows, a separation of concerns principle was enforced, decoupling the ML code (maintained by scientists) from the operational code (maintained by the deployment team in a separate repository) responsible for automating Kubeflow pipelines via CI/CD. ML code repositories primarily use Python and R, while operational code is standardized in YAML and Python templates defining Kubeflow components and CI/CD workflows. Both are organized under consistent folder structures to simplify maintenance. This approach was supported by robust version control and automatic testing within CI/CD pipelines.

For tracking business impact, the MLOps platform implemented procedures enabling monitoring and interpretability. Datadog[47] was used to collect metrics detailing model usage, such as prediction requests over time and resource utilisation. Logs were collected using Grafana Loki[48] and visualised using Grafana Dashboard[49]. Furthermore, system integrations and components were engineered to allow for the export of key metrics, artifacts, and uncertainty values associated with predictions, thereby fostering trust and adoption among end-users and domain experts. These

exported artifacts included cross-validation metrics, intermediate image masks, saliency maps, and model prediction uncertainty values.

Conclusion

As machine learning and artificial intelligence have become an intrinsic part of modern drug discovery efforts, the ability to deploy models and provide their outputs to scientists has become a critical capability for pharmaceutical companies. Such deployments usually present many challenges [50,51] such as scalability, reproducibility and traceability, thereby requiring dedicated tools and sometimes engineering teams to overcome them.

The MLOps platform presented here provides a concrete and novel solution to this challenge, and it is to our knowledge the first of its kind reported in the scientific literature for life sciences. Its ability to handle multiple data modalities across different stages of drug discovery makes it a very powerful and essential tool for Research & Development organizations, and it differentiates it from previously reported approaches in life sciences. Indeed, existing solutions only focus on narrow parts of the discovery process such as lead identification and/or optimization for small molecules [22], making them unfit to support the entire pre-clinical discovery process. Beyond handling multiple data modalities, our MLOps platform offers a holistic approach to operationalize models throughout the entire drug discovery process, integrating with a diverse and complex digital ecosystem and prioritizing a clear separation between ML and operational code to enhance reliability and focus. We achieved this desired flexibility by selecting

technologies that allow scientists to deploy any types of ML workflows and pipelines, including training and inference, batched or online.

To successfully cover the entire drug discovery process, we also invested in integration interfaces with specific components of the digital ecosystem used by scientists to interpret ML outputs in the appropriate scientific context. This contributed very strongly to the success of our MLOps platform, as it ensured accessibility of predictions and models throughout the entire organization. Coupled with observability tools, it also enabled us to assess the impact of each model on the discovery process, which is becoming increasingly important given the high costs of training and deploying cutting edge AI models, especially in the field of generative AI. This approach also enabled us to overcome the limitations of existing off-the-shelf MLOps solutions, which did not adequately meet our requirements. Many of these alternatives addressed only specific phases of the model life cycle, supported a limited range of data modalities, or lacked the scalability necessary to accommodate our diverse use cases.

Our harmonized MLOps framework for drug discovery facilitates the training and onboarding of scientists by reducing technological complexity across the enterprise environment. This harmonization also contributes to a lower operational footprint relative to alternative MLOps solutions that are narrowly tailored to specific use cases. Moreover, the establishment of a centralized engineering team responsible for model deployment has enhanced code reusability and the systematic sharing of knowledge, thereby accelerating the release and maturation of production models over time.

In addition, the separation of concerns between operational and ML code empowers scientists to focus on accelerating drug discovery processes rather than manage daily ML operations. The use of reproducible, shareable pipelines and code contributes to effective collaborations between scientists, an increasing trend worldwide which is sustained by platforms such as Hugging Face© [16,17]. Our key learnings from these approaches emphasize that a structured approach, starting with predefined pipeline architectures and fostering a culture of well-structured code, significantly streamlines the release of models in production. Continuous quality assurance through automated testing, robust version control, and ongoing performance monitoring are critical for maintaining reliability and enabling timely interventions on production workflows. Ultimately, fostering open communication and collaborative interactions is paramount to overcoming inherent complexities and preventing silos in the deployment of AI models.

Finally, we anticipate that the advance of complex and semi-autonomous AI-powered workflows such as lab-in-the-loop therapeutic antibody design [44] or fully agentic solutions will require more scalable and flexible MLOps platforms such as the one presented here. Indeed, to sustain these transformative approaches, it will become paramount to deliver AI inference at scale, and with great traceability. This will only be achieved thanks to adequate investment in foundational MLOps capabilities, relying on scalable, reliable and flexible technologies.

CRediT authorship contribution statement

Elina Koletou: Conceptualization, Methodology, Formal analysis, Investigation, Supervision, Funding acquisition, Project administration, Writing – original draft, Writing – review and editing. **Le Mu**: Software, Conceptualization, Methodology, Investigation, Supervision, Writing – review and editing. **Raya Stoyanova:** Software, Methodology, Investigation, Writing – original draft, Writing – review and editing. **Jorge Oswaldo Gomez Muñoz:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Krystian Kozieł:** Software, Methodology, Investigation, Writing – review and editing. **Paweł Cisło:** Software, Methodology, Investigation, Writing – review and editing. **Lorenzo Maffioli**: Software, Methodology, Investigation, Writing – review and editing. **Dariusz Adamczyk:** Software, Methodology, Investigation, Writing – review and editing. Jakub Witkowski **Małgorzata Bartosik:** Software, Methodology, Investigation, Writing – review and editing. **Małgorzata Widelicka:** Software, Methodology, Investigation, Writing – review and editing. **Marius Quadflieg:** Software, Methodology, Investigation, Writing – review and editing. **Olga Maksymiuk-Młynarczyk:** Software, Methodology, Investigation, Writing – review and editing. **Santiago Gonzalez:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Yun Yvonna Li:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Josep Arús-Pous:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Witold Przeciechowski:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Łukasz Rakoczy:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Marek Grzenkowicz:**

Software, Methodology, Investigation, Writing – review and editing. **Monika Dyrda:** Software, Methodology, Investigation, Writing – review and editing. **Alin Spinu:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **Michael Reutlinger:** Conceptualization, Software, Methodology, Investigation, Supervision, Writing – review and editing. **Fabian Schmich:** Conceptualization, Software, Methodology, Investigation, Writing – review and editing. **David Zilian:** Conceptualization, Methodology, Investigation, Writing – review and editing. **Lykke Pedersen:** Conceptualization, Methodology, Investigation, Writing – review and editing. **Daniel Butnaru:** Conceptualization, Investigation, Methodology, Supervision, Funding acquisition, Writing – review and editing. **Angelika Fuchs:** Conceptualization, Methodology, Supervision, Funding acquisition, Writing – review and editing. **Anna Bauer-Mehren:** Conceptualization, Investigation, Methodology, Supervision, Funding acquisition, Writing – review and editing. **Christophe Chabbert:** Conceptualization, Methodology, Formal analysis, Investigation, Supervision, Funding acquisition, Project administration, Writing – original draft, Writing – review and editing.

Declaration of competing interest

A patent application has been filed by Roche relating to the MLOps platform architecture and system integration strategies described in this manuscript. Beyond this patent application, the authors declare that they have no known competing financial interests or personal relationships with other people or organizations that could inappropriately influence or bias their work.

Funding Sources

The work is funded by Roche. Some authors are employees and stockholders at Roche.

Acknowledgements

We extend our sincere gratitude to Eric Keller and Rasmus Iten for their continuous support, and strategic guidance throughout the complexities of open sourcing compliance.

The foundation of our computational infrastructure relies heavily on open-source contributions; thus, we gratefully acknowledge the innovation and active maintenance provided by the Open Source Kubeflow community. We also thank the colleagues at Roche who initially established the connection and facilitated the adoption of this technology and particularly Krzysztof Romanowski.

For their crucial support regarding subject matter expertise and development, we thank the various colleagues from the Science & Research domain in the Digital Technology organization at Roche.

Finally, we appreciate the dedication and organizational support provided by all the scrum masters, Aleksandra Kremkowska, Marta Ozyra, Patrycja Gorka, Mariusz Mielniczuk, who ensured the efficient management and smooth execution of our project development workflows.

Data availability

The data and code that support the findings of this study are available in repositories and as supplementary material accompanying this article, promoting transparency and reproducibility.

Research Data: The statistics and formalized technical requirements collected during the initial phase of platform design are available as supplementary material:

- Supplementary Data Tables (PDF document)
- Supplementary Data (CSV file)

Code and Software: The operational strategies and architectural components described in this manuscript, which include reusable code and pipeline definitions, are publicly available. The associated code and software can be accessed via the following repositories:

- Operational code repository template: https://gitlab.com/roche/pred-mlops-papers/ops-repo-template
- ML code repository template: https://gitlab.com/roche/pred-mlops-papers/ml-repo-template

The provision of this software and code as research output maximizes the impact of this research by allowing colleagues the opportunity to download and cite the work. The research data and code are made available to allow reproducibility and align with community standards on best practices for research data management.

References

[1] Z. Du, H. Su, W. Wang, L. Ye, H. Wei, Z. Peng, I. Anishchenko, D. Baker, J. Yang,

The trRosetta server for fast and accurate protein structure prediction, Nat. Protoc. 16 (2021) 5634–5651. https://doi.org/10.1038/s41596-021-00628-9.

[2] Z.C. Drake, J.T. Seffernick, S. Lindert, Protein complex prediction using Rosetta, AlphaFold, and mass spectrometry covalent labeling, Nat. Commun. 13 (2022) 7846. https://doi.org/10.1038/s41467-022-35593-8.

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S.A.A. Kohl, A.J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A.W. Senior, K. Kavukcuoglu, P. Kohli, D. Hassabis, Highly accurate protein structure prediction with AlphaFold, Nature 596 (2021) 583–589. https://doi.org/10.1038/s41586-021-03819-2.

[4] D. Paul, G. Sanap, S. Shenoy, D. Kalyane, K. Kalia, R.K. Tekade, Artificial intelligence in drug discovery and development, Drug Discov. Today 26 (2021) 80–93. https://doi.org/10.1016/j.drudis.2020.10.010.

[5] O. Siméoni, H.V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, P. Bojanowski, DINOv3 | Research - AI at Meta, (2025). https://ai.meta.com/research/publications/dinov3/ (accessed September 26, 2025).

[6] D. Reidenbach, M. Livne, R.K. Ilango, M. Gill, J. Israeli, Improving Small Molecule Generation using Mutual Information Machine, Preprint Arxiv (2023).

https://arxiv.org/abs/2208.09016.

[7] G. Corso, H. Stärk, B. Jing, R. Barzilay, T. Jaakkola, DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking, Preprint Arxiv (2023). https://arxiv.org/abs/2210.01776.

[8] E. Nguyen, M. Poli, M.G. Durrant, B. Kang, D. Katrekar, D.B. Li, L.J. Bartie, A.W. Thomas, S.H. King, G. Brixi, J. Sullivan, M.Y. Ng, A. Lewis, A. Lou, S. Ermon, S.A. Baccus, T. Hernandez-Boussard, C. Ré, P.D. Hsu, B.L. Hie, Sequence modeling and design from molecular to genome scale with Evo, Science 386 (2024) eado9336. https://doi.org/10.1126/science.ado9336.

[9] J.P. Taylor-King, M. Bronstein, D. Roblin, The Future of Machine Learning Within Target Identification: Causality, Reversibility, and Druggability, Clin. Pharmacol. Ther. 115 (2024) 655–657. https://doi.org/10.1002/cpt.3158.

[10] Y. You, X. Lai, Y. Pan, H. Zheng, J. Vera, S. Liu, S. Deng, L. Zhang, Artificial intelligence in cancer target identification and drug discovery, Signal Transduct. Target. Ther. 7 (2022) 156. https://doi.org/10.1038/s41392-022-00994-0.

[11] M.Y. Lu, B. Chen, D.F.K. Williamson, R.J. Chen, M. Zhao, A.K. Chow, K. Ikemura, A. Kim, D. Pouli, A. Patel, A. Soliman, C. Chen, T. Ding, J.J. Wang, G. Gerber, I. Liang, L.P. Le, A.V. Parwani, L.L. Weishaupt, F. Mahmood, A multimodal generative AI copilot for human pathology, Nature 634 (2024) 466–473. https://doi.org/10.1038/s41586-024-07618-3.

[12] Z. Huang, F. Bianchi, M. Yuksekgonul, T.J. Montine, J. Zou, A visual–language foundation model for pathology image analysis using medical Twitter, Nat. Med. 29 (2023) 2307–2316. https://doi.org/10.1038/s41591-023-02504-3.

[13] D. Leff, C. Chapo, VBStaff, Why do 87% of data science projects never make it into production?, (2019). https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production/ (accessed December 4, 2024).

[14] R. Panikkar, T. Saleh, M. Szybowski, R. Whiteman, Operationalizing machine learning in processes, McKinsey & Company (2021). https://www.mckinsey.com/capabilities/operations/our-insights/operationalizing-machine-learning-in-processes (accessed December 4, 2024).

[15] U. Fayyad, T.C. Forbes, Why Most Machine Learning Applications Fail To Deploy, Forbes (2023). https://www.forbes.com/councils/forbestechcouncil/2023/04/10/why-most-machine-learning-applications-fail-to-deploy/ (accessed December 4, 2024).

[16] HuggingFaceInc., Hugging Face, (2022). https://huggingface.co/ (accessed December 4, 2024).

[17] M. Heikkilä, BLOOM: Inside the radical new project to democratize AI, MIT Technology Review (2022). https://www.technologyreview.com/2022/07/12/1055817/inside-a-radical-new-project-to-democratize-ai/ (accessed December 4, 2024).

[18] NVIDIA, NIM for Developers | NVIDIA Developer, (2025). https://developer.nvidia.com/nim?sortBy=developer_learning_library%2Fsort%2Ffeatured_in.nim%3Adesc%2Ctitle%3Aasc&refinementList%5Bcontent_type%5D%5B0%5D=NIM&hitsPerPage=12 (accessed September 26, 2025).

[19] S. Shankar, R. Garcia, J.M. Hellerstein, A.G. Parameswaran, Operationalizing

Machine Learning: An Interview Study, Preprint Arxiv (2022). https://arxiv.org/abs/2209.09125.

[20] A.K. Narayanappa, C. Amrit, Transfer, Diffusion and Adoption of Next-Generation Digital Technologies, IFIP WG 8.6 International Working Conference on Transfer and Diffusion of IT, TDIT 2023, Nagpur, India, December 15–16, 2023, Proceedings, Part I, IFIP Adv. Inf. Commun. Technol. (2023) 101–114. https://doi.org/10.1007/978-3-031-50188-3_10.

[21] J. Ryseff, B.F.D. Bruhl, S.J. Newberry, The Root Causes of Failure for Artificial Intelligence Projects and How They Can Succeed: Avoiding the Anti-Patterns of AI | RAND, 2024. https://www.rand.org/pubs/research_reports/RRA2680-1.html (accessed August 21, 2025).

[22] G.M. Ghiandoni, E. Evertsson, D.J. Riley, C. Tyrchan, P.C. Rathi, Augmenting DMTA using predictive AI modelling at AstraZeneca, Drug Discov. Today 29 (2024) 103945. https://doi.org/10.1016/j.drudis.2024.103945.

[23] V. Shah, X. Yu, Spotlight: Montai Builds a Multimodal AI Platform for Drug Discovery Using NVIDIA NIM Microservices | NVIDIA Technical Blog, (2024). https://developer.nvidia.com/blog/spotlight-montai-builds-a-multimodal-ai-platform-for-drug-discovery-using-nvidia-nim-microservices/ (accessed September 26, 2025).

[24] M.K.P. Jayatunga, W. Xie, L. Ruder, U. Schulze, C. Meier, AI in small-molecule drug discovery: a coming wave?, Nat. Rev. Drug Discov. 21 (2022) 175–176. https://doi.org/10.1038/d41573-022-00025-1.

[25] S.R. Choi, M. Lee, Transformer Architecture and Attention Mechanisms in Genome Data Analysis: A Comprehensive Review, Biology 12 (2023) 1033.

https://doi.org/10.3390/biology12071033.

[26] S. Zhang, R. Fan, Y. Liu, S. Chen, Q. Liu, W. Zeng, Applications of transformer-based language models in bioinformatics: a survey, Bioinform. Adv. 3 (2023) vbad001. https://doi.org/10.1093/bioadv/vbad001.

[27] J. Jiang, L. Chen, L. Ke, B. Dou, C. Zhang, H. Feng, Y. Zhu, H. Qiu, B. Zhang, G. Wei, A review of transformers in drug discovery and beyond, J. Pharm. Anal. (2024) 101081. https://doi.org/10.1016/j.jpha.2024.101081.

[28] S. Madan, M. Lentzen, J. Brandt, D. Rueckert, M. Hofmann-Apitius, H. Fröhlich, Transformer models in biomedicine, BMC Méd. Inform. Decis. Mak. 24 (2024) 214. https://doi.org/10.1186/s12911-024-02600-5.

[29] E.P. and of the Council, EUR-Lex - 02016R0679-20160504 - EN - EUR-Lex, (2016). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1532348683434 (accessed September 26, 2025).

[30] E.P. and of the Council, Regulation - EU - 2024/1689 - EN - EUR-Lex, (2024). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R1689 (accessed September 26, 2025).

[31] V. Makarov, C. Chabbert, E. Koletou, F. Psomopoulos, N. Kurbatova, S. Ramirez, C. Nelson, P. Natarajan, B. Neupane, Good machine learning practices: Learnings from the modern pharmaceutical discovery enterprise, Comput. Biol. Med. 177 (2024) 108632. https://doi.org/10.1016/j.compbiomed.2024.108632.

[32] K. Salama, J. Kazmierczak, D. Schut, Practitioners guide to MLOps: A framework for continuous delivery and automation of machine learning, Google Could White Paper

(2021).

[33] K. Romanowski, Project Lightning Talk: Kubeflow Helm Chart - Krzysztof Romanowski, Maintainer - YouTube, in: Kubecon and CloudNativeCon Europe 2025, 2025. https://www.youtube.com/watch?v=NkOV4_JV4t4 (accessed September 26, 2025).

[34] K. Romanowski, GitHub - kromanow94/kubeflow-manifests: A repository for Kubeflow Kustomize manifests, 2025. https://github.com/kromanow94/kubeflow-manifests.

[35] A. Tetelman, KServe Providers Offering NIM Inference in Clouds and Data Centers | NVIDIA Blog, (2024). https://blogs.nvidia.com/blog/kserve-nim-inference/ (accessed September 26, 2025).

[36] S. Agarwal, Y. Tang, A. Tetelman, R. Esker, Empower conversational AI at scale with KServe | Red Hat Developer, (2024). https://developers.redhat.com/articles/2024/03/15/empower-conversational-ai-scale-kserve (accessed September 26, 2025).

[37] A. Munteanu, Deploy GenAI applications with Canonical's Charmed Kubeflow and NVIDIA NIM | Ubuntu, (2024). https://ubuntu.com/blog/deploy-genai-applications-with-nvidia-nim-and-charmed-kubeflow (accessed September 26, 2025).

[38] D. Sun, Y. Liu, The journey to build Bloomberg's ML Inference Platform Using KServe (formerly KFServing) | Bloomberg LP, (2021). https://www.bloomberg.com/company/stories/the-journey-to-build-bloombergs-ml-inference-platform-using-kserve-formerly-kfserving/ (accessed September 26, 2025).

[39] F. Pachinger, M. Maurer, Building a Machine Learning Pipeline with Cisco Intersight and Kubeflow - Cisco Blogs, (2022). https://blogs.cisco.com/developer/machinelearningpipeline01 (accessed September 26, 2025).

[40] A. Mendez, Production-ready object detection model training workflow with HPE Machine Learning Development Environment | HPE Developer Portal, (2023). https://developer.hpe.com/blog/production-ready-object-detection-model-training-workflow-with-hpe-machine-learning-development-environment/ (accessed September 26, 2025).

[41] P.B. Cox, R. Gupta, Contemporary Computational Applications and Tools in Drug Discovery, ACS Med. Chem. Lett. 13 (2022) 1016–1029. https://doi.org/10.1021/acsmedchemlett.1c00662.

[42] ©_2024_The_Kubeflow_Authors, Build a Pipeline | Kubeflow, Kubeflow.Org (2024). https://www.kubeflow.org/docs/components/pipelines/legacy-v1/sdk/build-pipeline/ (accessed December 9, 2024).

[43] S.M. Paul, D.S. Mytelka, C.T. Dunwiddie, C.C. Persinger, B.H. Munos, S.R. Lindborg, A.L. Schacht, How to improve R&D productivity: the pharmaceutical industry's grand challenge, Nat. Rev. Drug Discov. 9 (2010) 203–214. https://doi.org/10.1038/nrd3078.

[44] N.C. Frey, I. Hötzel, S.D. Stanton, R. Kelly, R.G. Alberstein, E.K. Makowski, K. Martinkus, D. Berenberg, J. Bevers, T. Bryson, P. Chan, Y. Chen, A. Czubaty, T. D'Souza, H. Dwyer, A. Dziewulska, J.W. Fairman, A. Goodman, J. Hofmann, H. Isaacson, A. Ismail, S. James, T. Joren, S. Kelow, J.R. Kiefer, M. Kirchmeyer, J.

Kleinhenz, J.T. Koerber, J. Lafrance-Vanasse, A. Leaver-Fay, J.H. Lee, E. Lee, D. Lee, W.-C. Liang, J.Y.-Y. Lin, S. Lisanza, A. Loukas, J. Ludwiczak, S.P. Mahajan, O. Mahmood, H. Mohammadi-Peyhani, S. Nerli, J.W. Park, J. Park, S. Ra, S. Robinson, S. Saremi, F. Seeger, I. Sinha, A.M. Sokol, N. Tagasovska, H. To, E. Wagstaff, A. Wang, A.M. Watkins, B. Wilson, S. Wu, K. Zadorozhny, J. Marioni, A. Regev, Y. Wu, K. Cho, R. Bonneau, V. Gligorijević, Lab-in-the-loop therapeutic antibody design with deep learning, bioRxiv (2025). https://doi.org/10.1101/2025.02.19.639050.

[45] Google, MLOps: Continuous delivery and automation pipelines in machine learning | Cloud Architecture Center | Google Cloud, (2024). https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (accessed November 12, 2025).

[46] Gravitee.io, API Gateway - Secure, Scalable, Open Source Solution | Gravitee, (2025). https://www.gravitee.io/platform/api-gateway (accessed September 26, 2025).

[47] Datadog, Cloud Monitoring as a Service | Datadog, (2025). https://www.datadoghq.com/ (accessed September 26, 2025).

[48] L. Grafana, Grafana Loki OSS | Log aggregation system, (2025). https://grafana.com/oss/loki/ (accessed November 26, 2025).

[49] L. Grafana, Grafana dashboards | Grafana Labs, (2025). https://grafana.com/grafana/dashboards/ (accessed November 26, 2025).

[50] L. Wossnig, N. Furtmann, A. Buchanan, S. Kumar, V. Greiff, Best practices for machine learning in antibody discovery and development, Drug Discov. Today 29 (2024) 104025. https://doi.org/10.1016/j.drudis.2024.104025.

[51] A. Bender, I. Cortés-Ciriano, Artificial intelligence in drug discovery: what is realistic,

what are illusions? Part 1: Ways to make an impact, and why we are not there yet, Drug Discov. Today 26 (2021) 511–524. https://doi.org/10.1016/j.drudis.2020.12.009.