Drug Discovery Machine Learning Systems For Accelerating Idea Hypothesis To Production Decisions

Carsten Stahlhut Jesper Ferkinghoff-Borg Kang Li Kilian W. Conde-Frieboes Vanessa I. Jurtz Christian Vind Kristoffer Balling Søren B. Padkjær
Novo Nordisk A/S *
ctqs@novonordisk.com

Abstract

Drug discovery is a highly iterative process calling for continual learning, where explorative steps foster the base information for the more directed molecule optimization utilized in exploit steps. Machine learning driven techniques have proven a promising direction to assist in this guidance. Yet, minimal attention has been given to the challenges and needs in securing the ability to learn across drug discovery projects while still being able to iterate quickly through ideation phase to testing and deployment steps. In this contribution we discuss our experiences and considerations of integrating an MLOps setup in the drug discovery process. We focus specifically on capturing the idea generation process through experimental tracking while still being able to minimize development time from experimentation to actual deployment.

1 Introduction

Developing new medicine is an inherently challenging and expensive process that typically requires between 10-20 years in development and approval time before reaching the market [1, 2]. This prolonged time is among other things a consequence of a complicated search for a suite of desired functional properties in an extremely large chemical space. Even if the search is restricted to natural amino acids only, the chemical space is in the order of 10^{40} for peptides of a moderate size of 30 amino-acids.

To accommodate a structural search in such a large space, with only a tiny subspace satisfying the desired functional properties, we utilize several machine learning techniques, including representation learning and predictive models of the molecules' function to active learning models controlling the actual search strategy.

However, relying on several machine learning (ML) model steps in itself have a direct consequence of the operation of the system and require well defined interfaces from data acquisition process deployment and monitoring of the ML system behaviour when running in production. Several excellent ML driven drug discovery examples have demonstrated how to use ML models to predict optimized functions such as [2, 3]. However, minimal attention has been given to the full eco-system around reproducible ML system development entailing tracking, deployment as well as ensuring that the system is operated as expected. While setting up a PoC example of a ML system is typically easy using various boiler-plate code from the web, it is much harder to set up an integrated model without the surrounding infrastructure. In fact recent computational reproducibility analysis provided by [4] indicates that out of 9625 evaluated Jupyter notebook workflows the authors were only able to reproduce the results in 245 of the cases with an associated biomedical publication despite provided Github code. These numbers speak their own words about the challenges in providing reproducible data science work.

*Novo Nordisk A/S, R&ED, AI & Digital Research, Novo Nordisk Park 1, DK-2760 Måløv.

In this paper we formulate our ML system from a lab perspective both in the sense of how to treat experiments in the lab as well as how to scale up the experiments to accelerate scientific hypotheses, supported by our in-silico laboratory testing. Driving high-throughput experiments in a physical laboratory require well-defined protocols in order to ensure high quality data at scale. The equipment needs to operate as expected, and if not this can be detected through different alerts. In a MLOps context we view an in-silico experiment to be in direct correspondence to an in-vitro or an in-vivo experiment with respect to clarifying the material and operational requirements needed to ensure full reproducibility.

2 Drug discovery in an MLOps context

To ensure the acceptance and support from the core business with respect to the commitment and investment needed for implementing good MLOps principles such as [5, 6], we have found it very useful to emphasize the similarities between the objectives, challenges and stages pertaining to model development with those pertaining to in-vitro assay development. It is in the model ideation stage we outline our hypothesis and associated constraints in order to e.g. navigate in an enormous chemical space in the context of molecule optimization. At this critical stage we are introducing inductive biases as a consequence of the constraints being tested in experimentation which form the basis for the further development and eventual fine-tuning prior to the final deployment. While at the first stage the inductive biases might seem too hard and fluffy to record, they define the later boundaries of the search space and thus the applicability of the selected search strategy at hand. More importantly, it is at this stage we foster our later ability to secure the data required to enable transfer learning between projects. These considerations are no different than those related assay development to measure one specific property at hand. Thus, we see an ML model as a dedicated instance of an assay, with the main difference that the model here is an in-silico assay rather than a physical one. Furthermore, we make the analogy that a model definition essentially corresponds to a protocol definition of how to run a specific assay. Similarly, the ML system and its operation (MLOps) in this context are comparable to the core infrastructure that ensures that the assay runs accordingly to the protocol (model) definition.

2.1 MLOps architecture

As part of the early drug discovery phase our main focus is to capture and standardize the information in the model development phase such as how objectives are changing as part of the project progression, from e.g. its primary focus on receptor binding to properties related to stability, formulation, process optimization etc. We are naturally interested in optimizing the full development cycles and thus the most fundamental step is to record this data such that it is transparent how decisions were made in projects and how data can be utilized for transfer learning. This also provides guidance on quantitative measures of the performance of the ML system. To ease and support a standardized setup, designed to circumvent the well-known problem of maintaining boiler-plate code [5, 7], we propose that ML/data scientists comply to the use of dedicated templates, as indicated in Figure 1. Through the templates we facilitate the adoption of our scientists to different ML system abstractions, including ETL (extract, transfer, load process), model class, MLProject configuration file and package specification file. These abstractions significantly reduce the large code refactoring steps that are typically needed as part of packaging a development-stage model to deployment.

A number of open source templates and packages exist to help standardize the structure, e.g. `cookiecutter` [9] can be quite suitable for setting up well-defined templates. However, the templates in itself serve a potential risk of introducing more boiler-plate code and increasing maintenance efforts as the templates evolve over time with new functionality or bug fixes introduced and thus require to be maintained in itself. In order to propagate these changes and keep existing ML systems up to date if already based on a previous template version, we need a consistent schema for this. To fight back the maintenance overhead, open-source tools such as `cruft` can be utilized [10], which exactly try to automate this procedure.

2.2 Experimentation - enabling scientists in a controlled environment

We find that the transparency of the performance of a machine learning model system to be one of the fundamental steps towards building trust in the models. It is in this initial step that we augment our

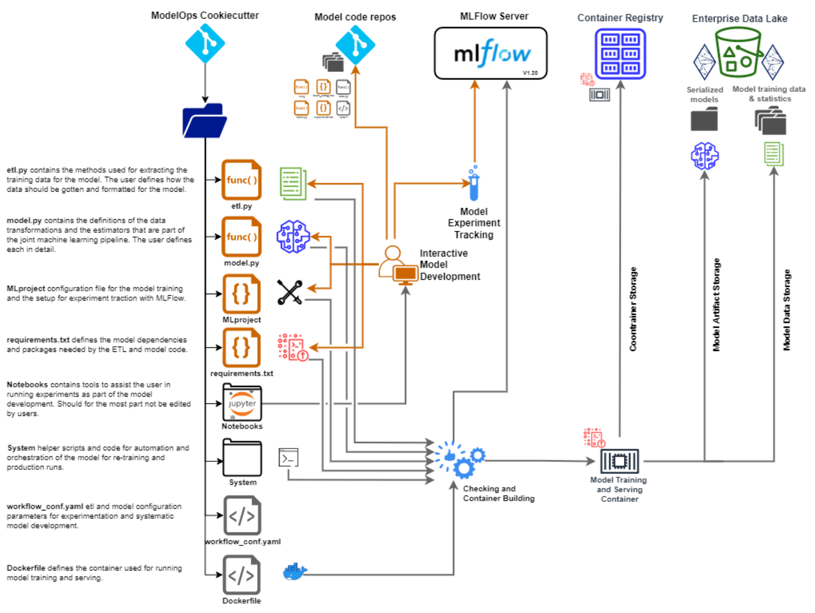

ModelOps Cookiecutter
Model code repos
MLFlow Server
mlflow
V1.20
Container Registry
Enterprise Data Lake
Serialized models
Model training data & statistics
etl.py contains the methods used for extracting the training data for the model. The user defines how the data should be gotten and formatted for the model.
func( )
etl.py
model.py contains the definitions of the data transformations and the estimators that are part of the joint machine learning pipeline. The user defines each in detail.
func( )
model.py
MLproject configuration file for the model training and the setup for experiment traction with MLFlow.
MLproject
requirements.txt defines the model dependencies and packages needed by the ETL and model code.
requirements.txt
Notebooks contains tools to assist the user in running experiments as part of the model development. Should for the most part not be edited by users.
jupyter
Notebooks
System helper scripts and code for automation and orchestration of the model for re-training and production runs.
System
workflow_conf.yaml etl and model configuration parameters for experimentation and systematic model development.
workflow_conf.yaml
Dockerfile defines the container used for running model training and serving.
Dockerfile
Model Experiment Tracking
Interactive Model Development
Coontrainer Storage
Model Artifact Storage
Model Data Storage
Checking and Container Building
Model Training and Serving Container

Figure 1: Architecture of model template outlining which component the users are expected to customize to fit for the purpose (orange) with pre-defined schema to benefit of the model system interfaces with experiment tracking tools such as MLFlow [8], building of model containers with associated registry and data storage layer in internal data lake to obey the data security policies.

machine learning system with well-defined data pipelines and machine learning pipelines to ensure reproducible results. Having full lineage from early experimentation of the model architecture and parameter configuration to later consolidation is a prerequisite for identifying potential problems or improvements to the models, as highlighted by [5]. In the development phase of a ML system we seek to foster flexibility to the ML/data scientist such that new state-of-the-art models, packages, alternative data transformation steps etc. can be tested easily. However, we also seek reproducible experiments and this challenges the robustness of the setup as well as the turn-around time for how quickly a stable environment can be provided to the scientist. As indicated in Figure 1, our scientists are at this early stage often prototyping in a Jupyter notebook like format. Here, we enable the container and provide preconfigured notebooks and scripts for performing model training and serialization. At each experiment run we enable model parameter logging, data versioning, code versioning and model packaging. Since the training examples constitute an integrated part of several models, including Support Vector Machines (SVM) and Gaussian Processes (GP), we also serialize our models back to our enterprise data lake to ensure a proper data access policy entailing the automatic capture of e.g. sensitive data.

2.3 Fair benchmarking

One of the fundamental requirements for model development and reasoning of the models is to secure fair benchmarks with consistent treatment of train, development and test data. Whilst this should be obvious, we often find that individual data scientists are tempted to do their own splitting into train, development, and test data to be able to shoe horn a quick analysis decision. This neglects the importance of full reproducible experiments as well as the much more important goal of being able to deliver the data to the foundation for continual learning across projects. To prevent ad-hoc model validation, a number of backtesting functionalities is provided to the scientists through the framework as demonstrated in Figure 2. Here, molecules variants are recorded and grouped in their associated design rounds to ensure fair model comparison. While this setup facilitates fair benchmarking for our predictive models it is not well-suited for active learning models, as challenger models will be forced to use the training trajectories from the current production model. Thus, for active learning algorithms benchmarking we relapse to simulations with benchmarks directed towards the minimum number of designs needed to reach e.g. some x-fold improvement.

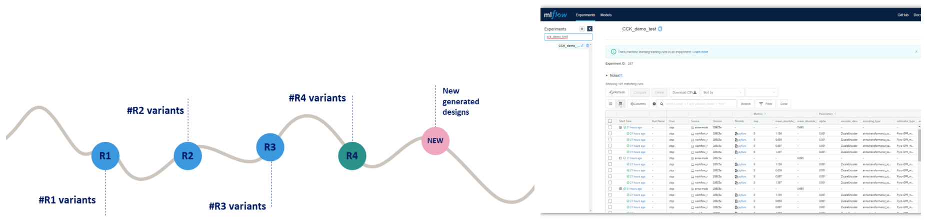


Figure 2: New ML models are consistently backtested. In each round the ETL controls that only train and development data is loaded from round R1 molecules and with R2 as test data. Performance is tracked in MLflow, A.

2.4 ML system deployment and monitoring

To support continuous integration and deployment (CI/CD) of our ML systems we have enabled daily scheduled jobs designed to both ensure that current model environments can be build from locked package versions as well as from scratch with newest package releases. Similarly, our scheduled jobs contain retraining and performance verification through standardized variant design backtesting as illustrated in Figure 2. We utilize the backtesting strategy on a daily basis to detect and learn if new challenger models are stepping up, e.g. due to newly added data. If the ML system passes the validation criteria it is deployed with a model API end point. Models can be registered and deployed through our MLFlow server and by utilizing the MLFlow build-in deployment functionalities. In practice, we typically utilize the Domino Enterprise MLOps platform [11] to deploy our models here, which empowers the ML system administrator with easy control of access control level as well as scaling up/down compute resources. While accessing a model via an API entry point might be suitable for data scientists, it is typically not enough for targeting non-data scientists as they generally require some sort of a GUI to utilize the ML system as an integrated part of their decision process. Depending on the maturity of the application and number of dedicated users we typically embed our model APIs into a `streamlit` application. Such application allows us to quickly iterate with our users on how they would like to work with the models when designing new compounds.

Through user statistics from our model APIs / applications we are utilizing tools such as `Graphite` [12] and Domino Enterprise MLOps platform built-in user statistics to keep track of which functionalities the users are accessing. Additionally, resource statistics are provided. For some of our production models, deployed on our internally hosted Domino Enterprise MLOps platform, we are utilizing the model monitoring capabilities to keep track of whether the deployed ML systems are being used as intended and whether potential problems are arising, such as the presence of data drifting. While detected data drifts can be configured to trigger a retraining of the ML-system, we have not enabled this yet as in most of our setups this is picked up from our scheduled jobs which entail the automatic retraining of a suite of challenger models as indicated above.

3 Conclusion

Utilizing ML systems as an integrated part of the drug discovery process can provide improved drug candidates quicker to the patients. We advocate for adhering to the best MLOps principles as early as possible in the design process. This should be done already at the idea generation phase. At this stage basic assumptions and decisions are made with respect to both data utilization and model design, which will impact the overall performance of the ML system. Through the usage of `cruft` [10], we provide ML templates that are easy to maintain and to apply in new projects. One of the largest challenges in incorporating standardized MLOps setups across research projects and work processes is to change the habits of the scientists, despite their wishes to have a better model framework for collaboration. Here, we argue that a prerequisite for such a data scientist based collaboration is to have a framework that ensures full transparency and reproducibility of the data scientific experiments. To mitigate old habits investment in teaching and on-boarding new colleagues is needed. Furthermore, we argue that it is the experiment tracking performance metrics together with the different ML model architecture parameters that build the foundation for enabling transfer learning across all the various drug discovery pipelines. We see this as an absolutely mandatory step to realize the full potential of modern machine learning and AI in the Pharma industry.

Acknowledgments and Disclosure of Funding

We thank Simone Fulle, Monica Jane, Paolo Marcatili, and Sune Askjær for fruitful discussion and maturation of the framework to ensure applicability of multi-modality use cases.

References

[1] phrma.org. Modernizing drug discovery, development, 3 2016.

[2] Alex Zhavoronkov, Yan A Ivanenkov, Alex Aliper, Mark S Veselov, Vladimir A Aladinskiy, Anastasiya V Aladinskaya, Victor A Terentiev, Daniil A Polykovskiy, Maksim D Kuznetsov, Arip Asadulaev, et al. Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nature biotechnology*, 37(9):1038–1040, 2019.

[3] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702.e13, 2020.

[4] Sheeba Samuel and Daniel Mietchen. Computational reproducibility of jupyter notebooks from biomedical publications. *arXiv preprint arXiv:2209.04308*, 2022.

[5] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28, 2015.

[6] Khalid Salama, Jarek Kazmierczak, and Donna Schut. Practitioners guide to mlops: A framework for continuous delivery and automation of machine learning. *Google Could White paper*, 2021.

[7] Martin Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[8] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018.

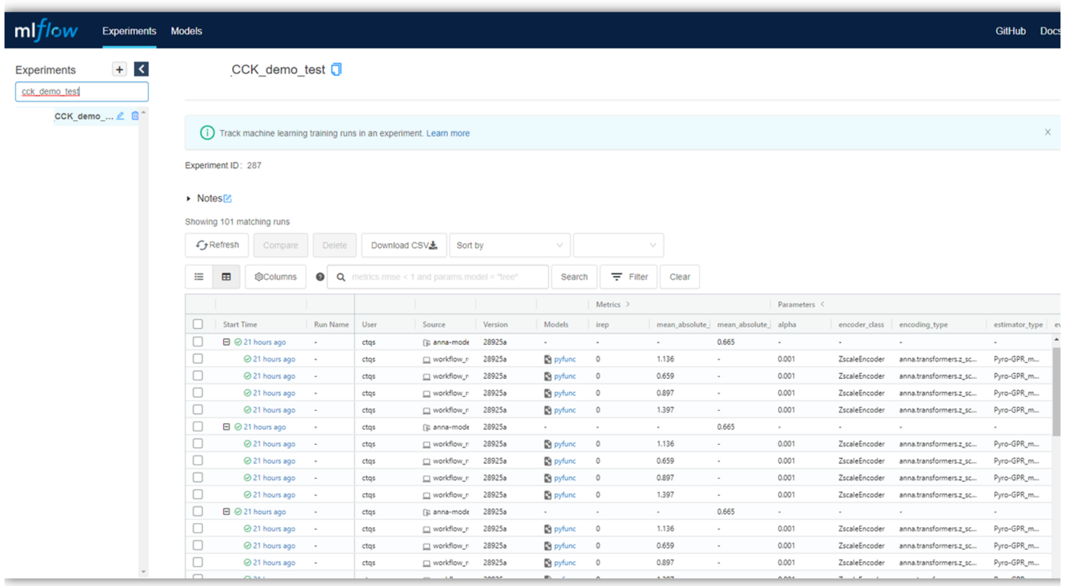
[9] Audrey Feldroy. Cookiecutter, 6 2022.

[10] Timothy Crosley. Cruft, 8 2022.

[11] Domino Data Lab. Domino's Enterprise MLOps Platform, 2022.

[12] Chris Davis. Graphite, 2022.

A Appendix



CCK_demo_test

Track machine learning training runs in an experiment. Learn more

Experiment ID: 287

Notes

Showing 101 matching runs

						Metrics			Parameters			
Start Time	Run Name	User	Source	Version	Models	irep	mean_absolute_	mean_absolute_	alpha	encoder_class	encoding_type	estimator_type
⊟ 21 hours ago	-	ctqs	anna-mode	28925a	-	-	-	0.665	-	-	-	-
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	1.136	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	0.659	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	0.897	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	1.397	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
⊟ 21 hours ago	-	ctqs	anna-mode	28925a	-	-	-	0.665	-	-	-	-
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	1.136	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	0.659	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	0.897	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	1.397	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
⊟ 21 hours ago	-	ctqs	anna-mode	28925a	-	-	-	0.665	-	-	-	-
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	1.136	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	0.659	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...
21 hours ago	-	ctqs	workflow_r	28925a	pyfunc	0	0.897	-	0.001	ZscaleEncoder	anna.transformers.z_sc...	Pyro-GPR_m...

Figure 3: Higher resolution of experiment tracking example given in Figure 2 for back-testing of variant design rounds.