2022 International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC) | 978-1-6654-9604-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICMACC54824.2022.10093671

Comparative Evaluation of Machine Learning Development Lifecycle Tools

Janvi Prasad
Dept. of Computer Science and Engg.
Vellore Institute Of Technology
Vellore, India
janvi.prasad@gmail.com

Arushi Jain
Dept. of Computer Science and Engg.
Vellore Institute Of Technology
Vellore, India
arushijain1479@gmail.com

Ushus Elizabeth Zachariah
Dept. of Computer Science and Engg.
Vellore Institute Of Technology
Vellore, India
ushus@vit.ac.in

***Abstract*—The ML development lifecycle is the SDLC equivalent of Machine Learning. While the ML code is at the core of a real-world ML production system, it frequently represents only 5% or less of the system's entire code. This study examines and contrasts the technologies utilised in the machine learning development lifecycle and focuses largely on the distinction between ML programming and ML development. According to Forrester Research, AI adoption is ramping up. 63% of business technology decision makers are implementing, have implemented, or are expanding use of AI. The main motivation behind this research is that the majority of the organizations do not have ML/AI solutions that have gone beyond the PoC / PoV stage, ML code in Jupyter notebooks cannot be distributed, and AI/ML solution deployment at scale is a challenge.**

Machine learning services are evolving at a dizzying rate, opening up a variety of opportunities for on-field applications, especially for brands and businesses with the infrastructure and resources required to integrate ML into their operational structures as a decision-making fulcrum. Nearly 65% of stock market swings may be predicted by Azure Machine Learning. By incorporating ML into its operational framework, Amazon has successfully decreased the "click-to-ship" time by 225%. Breast cancer can be identified with 89% accuracy using Google's Deep Learning. Thus, these commercial tools for ML life cycle support have been compared and a conclusion about which tool is most suitable is drawn.

***Keywords*—Machine Learning lifecycle, MLOps, Software Development lifecycle, MLFlow.**

I. Introduction

Writing the computer code necessary for software to run is known as software programming. The terms used in the field of computer technology is frequently overlapping and difficult to understand. Software development and software programming are not the same thing. Development is the actual design of a program, whereas programming is the execution of development's instructions.

To flesh it out, let's look at the core traits of a computer program, which is essentially a set of instructions that helps a computer perform a specific task:

- It makes sure the instructions are carried out successfully.
- It guarantees that the instructions are performed in the correct order.
- Explains whether the input (data) provided is accurate or insufficient and provides the outcome accordingly.
- It is written in a high level language.

In contrast, the SDLC is a process a software corporation will employ when working on a software project. It consists of a comprehensive plan explaining how to develop, maintain, replace, and alter or enhance certain software. A strategy for enhancing both the general software development process and the quality of the finished product is outlined in the life cycle.

ML programming is a minor component of the entire ML Development Life Cycle, much as "software development" or "programming" is a minor component of the broader Software Development Life Cycle. Machine learning is the practise of programming systems particularly to support self-improvement and learning. Designing methods for automatic data collection and utilisation by a system to learn more is the aim of machine learning.

These data science projects follow a cyclical process called the machine learning life cycle. It outlines each step to take in order to create projects based on artificial intelligence and machine learning (AI). Data preparation, model development, and deployment are the three primary processes that make up the life cycle of an ML project. These three elements are necessary for producing high-quality models. Therefore, to support all aspects of the ML Development Life Cycle, there is a need to have tools that can automate tasks at the different stages of the development life cycle.

There are numerous cloud platforms and frameworks that can manage the machine learning lifecycle. Currently, include Algorithmia, Amazon SageMaker, Azure Machine Learning, Domino Data Lab, the Google Cloud AI Platform, HPE Ezmeral ML Ops, Metaflow, MLflow, Paperspace, and Seldon. In this paper, first, the phases of the Software Development Lifecycle and Machine Learning Development Lifecycle are examined, and then some key features of ML Development Lifecycle tools are discussed. Further, MLFlow, which is an open source tool that facilitates the ML Development Lifecycle is explored, and finally, some of the best-performing and most popular tools by Amazon Web Services, Microsoft Azure and Google Cloud are compared and analysed based on how well they support each stage of the development process.

II. Background

A. Software Development Lifecycle

A typical software development life cycle has 6 stages:

1) Planning and Requirement Analysis

The requirement analysis step of the SDLC is crucial, and the data it generates is utilised to define the basic project strategy and conduct technical, operational, and financial product feasibility assessments. The study's conclusion lists the various methods that can be employed to complete the project successfully and with the fewest risks.

2) Defining Requirements

Following the conclusion of the requirement analysis, the following step is accurately defining, documenting, and securing customer or market analyst approval for the product needs that will be designed and developed throughout the project life cycle. This is accomplished using a Software Requirement Specification (SRS) document.

3) Designing the Product Architecture

The design strategy for the product architecture is developed, documented, and approved in a DDS - Design Document Specification based on the criteria given in the SRS. All of the product's architectural components, as well as how communication and data flow are represented, are explicitly defined by the design approach.

4) Building or Developing the Product

The actual development and construction of the product takes place in this stage of the SDLC. The programming code is generated according to the According to Design Document Specification (DDS). To create the code, developers employ programming tools like compilers, interpreters, debuggers, etc. The sort of software being produced affects the choice of programming language.

5) Testing the Product

This stage is regarded as a subset of all stages because testing operations are often integrated into all SDLC stages in contemporary models. However, this stage only relates to the product's testing phase, during which product flaws are discovered, monitored, corrected, and retested until the product satisfies the SRS document's quality requirements.

6) Deployment in the Market and Maintenance

Once the product has undergone testing and is ready for deployment, it is formally released in the relevant market. The product rollout may take place in stages depending on corporate strategy. After the product has been tested in a small market, it may then be launched to the public with ongoing maintenance, either as is or with proposed improvements, depending on the feedback.

B. ML Development Lifecycle

Machine learning engineers frequently experiment with new datasets, models, software libraries, and tuning parameters in order to optimise and improve the model accuracy since the performance of the model is fully dependent on the input data and the training process.

The machine learning development lifecycle consists of the following steps:

1) Building the machine learning model

This stage involves selecting the model type based on the application. Additionally, it discovers the model's application during the model learning phase, allowing for proper model construction in accordance with an intended requirements. There exist diverse machine learning models, such as classification, regression, clustering, and reinforcement learning models, as well as supervised and unsupervised models.

2) Data Preparation

Machine learning can be used with a wide range of data. Many different sources, including businesses, pharmaceutical companies, IoT devices, enterprises, banks, hospitals, etc., can provide this data. A lot of data is provided during the machine learning stage because increase in the amount of data coincides with providing the desired results. This output data can be analysed or used as seed data for machine learning systems or applications in other domains.

3) Model Training

This stage's objective is to create a model using the supplied data. To determine model parameters, such as polynomial coefficients or weights of the ML model, which serve to lower the error for the given data set, a portion of the training data is utilized. The remaining data is then used to test the model. In order to enhance the model's performance, these two phases are frequently repeated.

4) Parameter Selection

It entails choosing the hyperparameters, which are training-related parameters. The performance of the model ultimately depends on these parameters as they control the effectiveness of the training procedure. They are absolutely necessary for the machine learning model to be developed successfully.

5) Transfer Learning

The aim of transfer learning is applying the knowledge that is learned while tackling one problem to another problem which is unrelated but nevertheless linked. Reusing machine learning models in different contexts has many advantages. Although a model cannot be immediately moved from one domain to another, it can be used as a starting point for training a model at a later stage. It consequently drastically cuts down on training time.

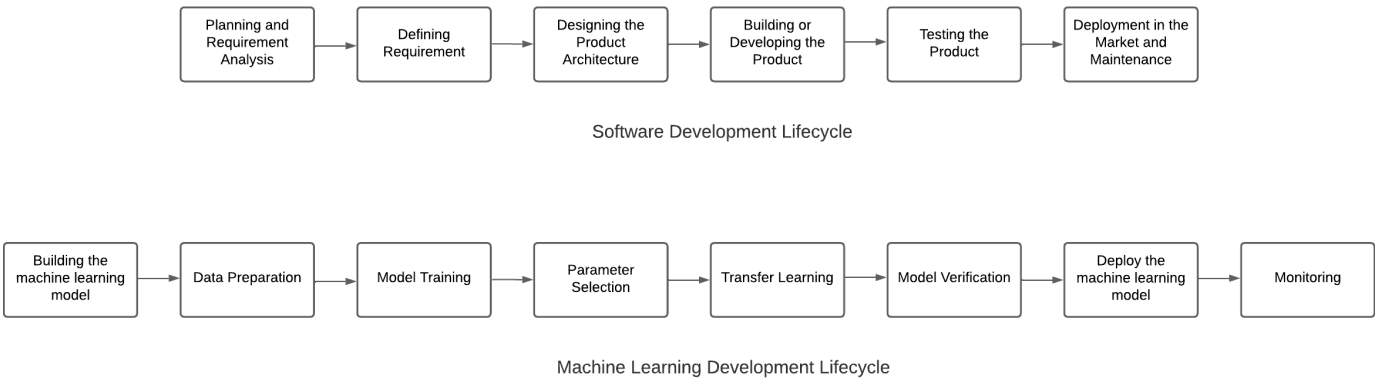

Fig. 1. SLDC vs ML Development Lifecycle

6) Model Verification

The trained model developed during the model learning stage is the input for this step, and the verified model that emerges from it provides enough information for users to decide whether the model is suitable for the application for which it is designed. As a result, the emphasis of this stage of the machine learning lifecycle is on figuring out whether a model works well when utilised with hidden inputs.

7) Deploy the machine learning model

At this stage of the machine learning lifecycle, machine learning models are incorporated into procedures and applications. The final objective of the stage is the effective operation of the model after deployment. The models need to be set up so that they may be utilised for inference, and they also need to be updated often.

8) Monitoring

It essentially involves the incorporation of safety precautions to guarantee the model's proper operation during the course of its lifetime. This can only be accomplished with the right management and updating.

III. Literature Review

[1] and [2] discuss how the Microsoft and Google AI platform, Azure and Google Cloud, respectively, support deep learning and building and deploying Artificial Intelligence solutions. In [3] and [4], the authors talk about how data scientists are being increasingly involved in software development teams. These teams can be considered the precursors of the data science and machine learning faculty intensive teams required for the ML development lifecycle.

[5] and [6] discuss the challenges faced in developing intelligent systems, [7] provides insights into some of the practices and capabilities of DevOps, [8] and [9] study and assess developers' work habits and how they develop software which lead to the need and evolution of the ML development lifecycle.

The concept of technical debt as it relates to ML systems is introduced in [10]. It highlights various ML-specific risk issues that system designers should keep in mind. [11] proposes a human-in-the-loop troubleshooting mechanism that focuses on fixing errors that arise in ML systems. [12] presents the TensorFlow Extended (TFX) platform to stabilize, simplify and standardize the process components of the ML development lifecycle. [13] emphasizes the need for Model Governance in production ML and [14] further investigates the advantages, challenges and applications which can help outline the overall structure of ML development.

The relationship between the stages of the software development life cycle is examined in [15] to ascertain whether machine learning favours any particular methodologies. [16] emphasizes the importance of accountable AI and presents Pandora, an evaluation method for describing and explaining system failures. [17] talks about explainable AI and its importance in trusting the behavior of ML models, by explaining otherwise overwhelmingly complex decisions and [18] proposes the Explainable AI (XAI) program to solve this problem and reu more effective management. [19] emphasises the value of datasheets for datasets in order to encourage the machine learning community to give transparency and accountability a priority. Finally, [20] and [21] talk about the applications of the ML development lifecycle in drug discovery and building facility cost analysis, respectively.

IV. Key Features of a tool

Tool must support all the following activities:

A. Data ingestion

Data ingestion is the process of acquiring and importing data for immediate use or storage in a database. To take something in or absorb something means to ingest it. Real-time data streaming and batch data ingestion are both options. During real-time data ingestion, each piece of data is imported as it is released by the source.

Data ingestion features and tools: Data ingestion tools have a variety of characteristics and capabilities, such as: Extraction, Processing, Data types, Data flow tracking and visualization, Volume, Security and privacy.

B. Data cleansing

Data cleaning is the process of removing erroneous, damaged, badly structured, duplicate, or incomplete data from a dataset. When combining various data sources, there are several possibilities for data to be duplicated or improperly categorised. Even though results and algorithms seem to be accurate, faulty data renders them unreliable. There is no definite method to define the specific steps in the data cleaning process because the procedures will vary from dataset to dataset. But in order to ensure that your data cleaning operation is carried out correctly each time, it is crucial to build a template for it. Five steps to carry out data cleaning:

- Eliminate redundant or irrelevant observations
- Correct structural issues
- Eliminate undesired outliers
- Handle missing data
- Validate and QA

C. Data validation

Checking the structure, quality, and integrity of data before using it for a business process is known as data validation. Results of data validation operations can give information for business intelligence, data analytics, or training a machine learning model. Additionally, it can be applied to guarantee data integrity for financial accounting or regulatory compliance. Numerous factors, such as data type, constraint, structure, consistency, and code validation, can be looked at while validating data. Each data validation technique aims to confirm that the data meets the requirements for being useful. Data quality and data validation are related. A component of measuring data quality is data validation, which makes sure that a given data collection is supplied with information sources that are of the highest caliber, are reliable, and are correct. The application procedures also incorporate data validation, which includes criteria for creating strong passwords and spell checking.

D. Model development

The process of developing a model is iterative; numerous models are generated, tested, and improved upon before a model that satisfies the required specifications is created. It may be necessary for subsequent modelling work to start the search where the initial model building began rather than where it ended. The term ”machine learning (ML) model lifecycle” refers to a process that starts with the identification of the source data and continues with the creation, deployment, and maintenance of the model. The entirety of the activities, such as ML Model Development and ML Model Operations, may be broadly divided into two groups.

Data exploration, model creation, adjusting model hyperparameters, and model selection with optimum performance are the steps in the lifespan of an ML model.

E. Model hyperparameter tuning

The control of a machine learning model's behavior requires hyperparameter adjustment. The predicted model parameters will yield less-than-ideal outcomes if the hyperparameters aren't properly tuned to minimize the loss function. This indicates that the model has more flaws. The accuracy or confusion matrix will actually be worse in reality.Parameters and hyperparameters must be distinguished in machine learning. For the provided data set, a learning algorithm learns or predicts the model parameters before updating these values over time. These variables are added to the model once learning is complete. One example of a parameter is each weight and bias in a neural network. On the other hand, the values of hyperparameters cannot be determined from the data because they are unique to the algorithm itself. To determine the model parameters, hyperparameters are employed. For a given data collection, various hyperparameter values result in various model parameter values.

F. Model deployment

The technical task of exposing an ML model to practical application is known as model deployment. The phrase is frequently used to refer to making a model available through real-time APIs. Though online inference on the cloud isn't always a necessary or even desirable answer, model deployment should be considered more broadly. A model must be successfully deployed into production before it can be used for making useful decisions. The impact of your model will be much diminished if you are unable to consistently derive useful insights from it.

G. Model validation and updation

The process of ensuring that a model effectively serves its intended function is known as model validation. This usually entails verifying that the model is accurate in the setting of its intended application. The process of updating a finite element model's parameters so that its eigenvalue and eigenvector predictions match data recorded during modal testing is known as ”model updating.”

V. MLFlow: An Open Source Tool for ML Life Cycle support

MLflow is influenced by current machine learning platforms, despite the fact that it is meant to be open in two ways:

A. Open interface

Any ML library, algorithm, deployment tool, or language may be used with MLflow because it has an open interface. Instead of just providing a small number of built-in features, it is built on REST APIs and fundamental data formats (for example, a model can be conceived as a lambda function) that can be used from a variety of tools. This also makes it simple to add MLflow to your current ML code so you can start benefiting from it right away and share code using any ML library that other employees of your company can execute.

B. Open source

If developers want to open source their code, sharing workflow steps and models between organizations is very simple thanks to MLflow's open format.

A central model registry, experimentation, reproducibility, and deployment are all managed through the open source platform MLflow. MLflow presently provides these four elements:

- MLflow Tracking: MLflow Tracking is an API and UI for logging parameters, code versions, metrics, and output files when running machine learning code so that users may later visualise them. Users are able to track parameters, metrics, and artifacts with just a few lines of code. Use MLflow Tracking in any setting (such as a standalone script or a notebook) to log results to local files or to a server, then compare various runs. Users may view and contrast the results of various runs using the web UI. The tools can be used by groups to compare the outcomes of various users.
- MLflow Projects: MLflow Projects provides a standardised approach for packaging reusable data science code. Each project essentially consists of a directory with the source code or a Git repository, and it uses a descriptor file to specify its dependencies and how to run the code. An MLflow project is defined in a simple YAML file called MLproject. The appropriate environment for the project will be automatically created and run by MLflow.
- MLflow Models: Machine learning models can be packaged using the MLflow Models protocol in a variety of "flavors," or forms. To assist you in deploying various model types, MLflow provides a number of tools. Each MLflow Model is maintained as a directory containing an arbitrary number of files and an MLmodel description file that lists the flavours it can be used with. MLflow offers tools for deploying a wide range of popular model types on various platforms.

C. Features of MLflow Model Registry

- Central Repository: To function as a central repository, register MLflow models with the MLflow Model Registry. An individual name, version, stage, and other metadata are all part of a registered model.
- Model versioning: Track updated versions of registered models automatically.
- Model Stage: To indicate the lifecycle of a model, preset or custom stages, such as "Staging" and "Production," are assigned to each model version. As activities that automatically log users, modifications, and extra metadata like comments, record new registration events or changes. This serves as a model for stage transitions.
- CI/CD Workflow Integration: Recording stage transitions and requesting, reviewing, and approving changes as part of CI/CD pipelines will improve control and governance.
- Model Serving: Rapidly deliver machine learning models via RESTful APIs on Databricks for online testing, dashboard updates, etc.

VI. Comparative evaluation of commercial tools for ML Life Cycle Support

AWS Sagemaker, Microsoft Azure Machine Learning and Google Tensorflow have been compared in the context of developing an ML pipeline to predict the presence of diabetes in patients. A Logistic Regression (LR) model has been built using the Diabetes 130-US hospitals dataset from the UCI Machine Learning Repository for data preparation. A reweighting algorithm has been applied to generate weights during training. The parameters selected are the patient's race, gender and age. Transfer learning has been applied with the pre-trained VGG convolutional neural network. The model has been verified based on the Disparate Impact fairness metric. The model has then been deployed using Tensorflow. The output of the model is monitored and if the diagnosis is insufficiently conclusive, the model is re-trained.

A. AWS Sagemaker

SageMaker should be used if a team consists of more developers than analysts and if they don't mind working on a platform that is undergoing an upgrade. However, this implies that they might probably produce better outcomes and be the first to use a tonne of new features. Sagemaker is perfect for businesses that create human-like conversation flows that call for more intricate intentions and prolonged discussions. If a single platform to build, train, deploy, and service your models is required, Sagemaker is a good choice. Sagemaker is the platform for someone who needs to handle a majority of machine learning-related jobs without stressing too much about the technical details.

B. Azure Machine Learning

If a simple and flexible building interface that aids enterprises in developing, testing, and producing advanced analytics based on the data is needed, use AzureML. When developing artificial intelligence (AI) applications, use Azure ML to analyze and act on data intelligently in real-time, enabling businesses to provide better results with more speed and efficiency. If you require support for a variety of frameworks, algorithms, and containers, Azure ML is the right choice. The use of ML pipelines and no-code designs expedites workflow creation and aids in considerably faster application scaling.

C. Google TensorFlow

TensorFlow, a powerful machine learning production, comes into play while discussing the MLOps platform. It is a machine learning platform that programmers use frequently to support the advancement of AI. One can create their own artificially intelligent programmes using this open-source software for a variety of applications that make use of machine learning. It's a useful toolkit that's even accessible on GitHub at TensorFlow for programmers of all skill levels who want to create the most powerful AIs.

VII. Conclusion

Given all the discussion about machine learning, the future will surely be data-driven. Organizations can employ machine learning capabilities to handle current business difficulties and prepare for future business opportunities. This paper examined three well-known platforms for the advancement of machine learning and artificial intelligence. Their features have been evaluated and contrasted to help you make a more informed decision about the best MLOps platform for your needs. All three Both are great options for developing and utilising machine learning, but each has benefits and drawbacks. The AWS Sagemaker framework is excellent for building simple models and quickly deploying them in the cloud. However, Azure ML may be a more flexible choice for predictive analytics. Winner: Azure ML

References

[1] M. Salvaris, D. Dean, and W. H. Tok, "Microsoft AI Platform," in Deep Learning with Azure. Springer, 2018.

[2] "Machine learning workflow," https://cloud.google.com/ml-engine/docs/tensorflow/ml-solutions-overview, accessed: 2018-09-24.

[3] Kim, M., Zimmermann, T., DeLine, R., Begel, A. (2016). The emerging role of data scientists on software development teams. Proceedings of the 38th International Conference on Software Engineering. https://doi.org/10.1145/2884781.2884783

[4] Kim, M., Zimmermann, T., DeLine, R., & Begel, A. (2018b). Data Scientists in Software Teams: State of the Art and Challenges. IEEE Transactions on Software Engineering, 44(11), 1024–1038. https://doi.org/10.1109/tse.2017.2754374

[5] M. Senapathi, J. Buchan, and H. Osman, "DevOps capabilities, practices, and challenges: Insights from a case study," in Proc. of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, 2018, pp. 57–67.

[6] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learning as a tool for software development," in Proc. of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2008.

[7] The Team Data Science Process," https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/, accessed: 2018-09-24.

[8] Hill, C., Bellamy, R.K., Erickson, T., & Burnett, M.M. (2016). Trials and tribulations of developers of intelligent systems: A field study. 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 162-170.

[9] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson, "How do scientists develop and use scientific software?" in Proc. of the 2009 ICSE workshop on Software Engineering for Computational Science and Engineering. IEEE Computer Society, 2009, pp. 1–8.

[10] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in NIPS, 2015.

[11] B. Nushi, E. Kamar, E. Horvitz, and D. Kossmann, "On human intellect and machine failures: Troubleshooting integrative machine learning systems." in AAAI, 2017, pp. 1017–1025.

[12] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc et al., "TFX: A tensorflow-based production-scale machine learning platform," in Proc. of the 23rd ACM SIGKDD. ACM, 2017, pp. 1387–1395.

[13] V. Sridhar, S. Subramanian, D. Arteaga, S. Sundararaman, D. Roselli, and N. Talagala, "Model governance: Reducing the anarchy of production ML," in USENIX. USENIX Association, 2018, pp. 351–358.

[14] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," Production Manufacturing Research, vol. 4, no. 1, pp. 23–45, 2016.

[15] Shafiq, Saad & Mashkoor, Atif Dorn, Christoph Egyed, Alexander. (2021). A Literature Review of Machine Learning and Software Development Life cycle Stages. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3119746.

[16] B. Nushi, E. Kamar, and E. Horvitz, "Towards Accountable AI: Hybrid Human-Machine Analyses for Characterizing System Failure", HCOMP, vol. 6, no. 1, pp. 126-135, Jun. 2018.

[17] D. S. Weld and G. Bansal, "Intelligible artificial intelligence," arXiv preprint arXiv:1803.04263, 2018.

[18] D. Gunning, "Explainable artificial intelligence (XAI)," Defense Advanced Research Projects Agency (DARPA), 2017.

[19] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. M. Wallach, H. D. III, and K. Crawford, "Datasheets for datasets," CoRR, vol. abs/1803.09010, 2018.

[20] Spjuth, O., Frid, J., & Hellander, A. (2021). The machine learning life cycle and the cloud: implications for drug discovery. Expert opinion on drug discovery, 16(9), 1071-1079.

[21] Gao, X., & Pishdad-Bozorgi, P. (2020). A framework of developing machine learning models for facility life-cycle cost analysis. Building Research Information, 48(5), 501-525.