ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

Review Article

Open Access

Deployment Strategies to make AI/ML Accessible and Reproducible

Ashish Bansal

USA

ABSTRACT
In recent years, big data and machine learning has been adopted in most of the major indus- tries and most startups are leaning towards the same. As data has become an integral part of all companies, ways to process them i.e. derive meaningful insights and patterns are essential. This is where machine learning comes into the picture. The emerging age of connected, digital world means that there are tons of data, distributed to various organizations and their databases. Since this data can be confidential in nature, it cannot always be openly shared in seek of artificial in- telligence (AI) and machine learning (ML) solutions. Instead, we need integration mechanisms, analogous to integration patterns in information systems, to create multi-organization AI/ML sys- tems.

There are many efficient machine learning systems to process the huge amount of data and based upon the task in hand, yield results in real-time as well. But these systems need to be curated and deployed properly so that the task at hand performs efficiently. Machine learning (ML) models are almost always developed in an offline setting, but they must be deployed into a production environment in order to learn from live data and deliver value.

A common complaint among ML teams, however, is that deploying ML models in production is a complicated process. It is such a widespread issue that some experts estimate that as many as 90 percent of ML models never make it into production in the first place. For the relatively few ML models that do make it to the production stage, ML model deployment can take a long time, and the models require constant attention to ensure quality and efficiency. For this reason, ML model deployment must be properly planned and managed to avoid inefficiencies and time-consuming challenges. This paper aims to provide you with information on the model deployment strategies and how you can choose which strategy is best for your application as well as how these strategies could be Reproduce to make faster deployments.

***Corresponding author**
Ashish Bansal, USA.

Received: December 05, 2022; **Accepted:** December 12, 2022; **Published:** December 26, 2022

Keywords: AI/ML, Model Deployment, MLOps, CI/CD, Containerization

Introduction

Machine learning (ML) has evolved from being an area of academic research to an applied field. According to a recent global survey conducted by McKinsey, machine learning is increasingly adopted in standard business processes with nearly 25% year-over-year growth and with growing interest from the general public, business leaders, and governments.

This shift comes with challenges. Just as with any other field, there are significant differences between what works in an academic setting and what is required by a real world system. Certain bottlenecks and invalidated preconceptions should always be expected in the course of that process. As more solutions are developed and deployed, practitioners report their experience in various forms, including publications and blog posts. Motivated by such reports and our personal experiences, in this study, we aim to survey the challenges in deploying machine learning in pro- duction1 with the objective of understanding what parts of the deployment process cause the mostdifficulties. First, we provide an overview of the machine learning deployment workflow. Second, we will provide various deployment strategies. Third, we discuss cross-cutting aspects that affect every stage of the deployment workflow: ethical considerations, law, end-users' trust, and security. Finally, we conclude with a brief discussion of potential solutions to these issues and further work.

Lifecycle of an ML model

The lifecycle of a machine learning model refers to the entire process that structures the whole data science or an AI project. It is similar to the software development life cycle (SDLC) but differs in a few key areas such as the use of real-time data to evaluate the model performance before deployment. A life cycle of the ML model or model development life cycle (MDLC) primarily has five phases as shown in Figure 1.

- Data collection
- Create model and training
- Testing and evaluation
- Deployment and production
- Monitoring

MLOps and Deployment Strategies

MLOps is generally a set of practices that enables ML Lifecycle. Its stitches machine learning and software applications together. Simply put, it is a collaboration between data scientists and the operations team that takes care of and orchestrates the whole ML lifecycle. The three key areas that MLOps focuses on are continuous integration, continuous deployment, and continuous testing.

Model deployment (release) is a process that enables you to integrate machine learning models into production to make decisions on real-world data. It is essentially the second last stage of the ML life cycle before monitoring. Once deployed, the model further needs to be monitored to check whether the whole process of data ingestion, feature engineering, training, testing are aligned properly so that no human intervention is required and the whole process is automatic.

But before deploying the model, one has to evaluate and test if the trained ML model is fit to be deployed into production. The model is tested for performance, efficiency, even bugs, and issues. There are various strategies one can use before deploying the ML model. Let us explore them. Below are various strategies and techniques for model deployment:

- Shadow evaluation
- A/B testing
- Multi Arm Bandits
- Blue-green deployment
- Canary testing
- Feature flag
- Rolling deployment
- Recreate strategy

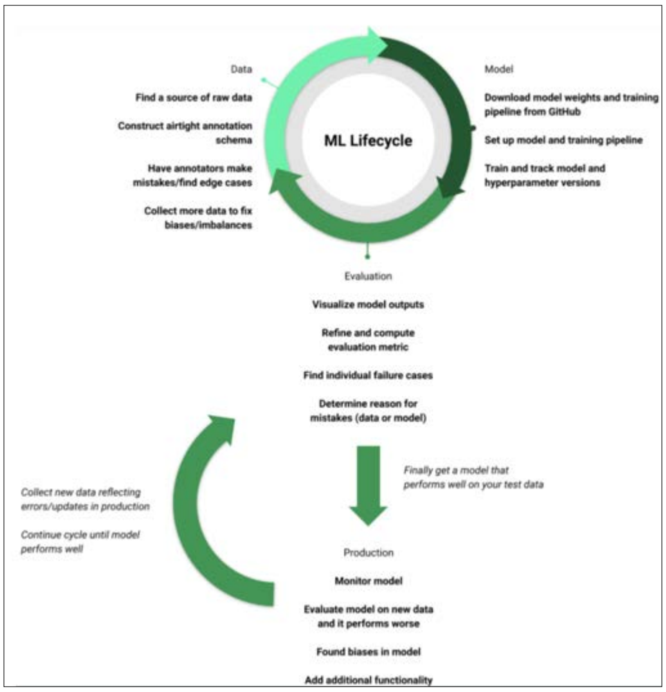
Data
Find a source of raw data
Construct airtight annotation schema
Have annotators make mistakes/find edge cases
Collect more data to fix biases/imbalances
ML Lifecycle
Model
Download model weights and training pipeline from GitHub
Set up model and training pipeline
Train and track model and hyperparameter versions
Evaluation
Visualize model outputs
Refine and compute evaluation metric
Find individual failure cases
Determine reason for mistakes (data or model)
Finally get a model that performs well on your test data
Collect new data reflecting errors/updates in production
Continue cycle until model performs well
Production
Monitor model
Evaluate model on new data and it performs worse
Found biases in model
Add additional functionality

Figure 1: ML Lifecycle

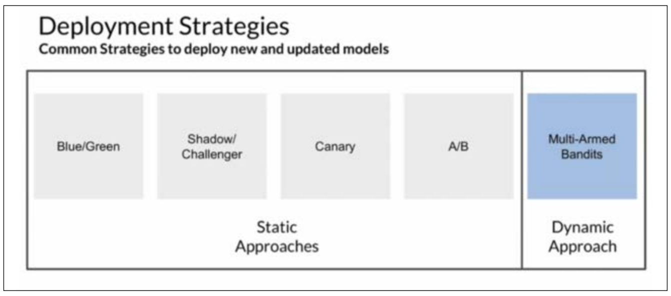

Figure 2: Deployment Strategies

These strategies can be broken down into two categories:

- **Static deployment strategies:** These are the strategies where the distribution of traffic or request are handled manually. Examples of this are shadow evaluation, A/B testing, Canary testing, Rolling deployment, Blue-green deployment et cetera.
- **Dynamic deployment strategies:** These are the strategies where the distribution of traffic or request are handled automatically. Example of this is Multi Arm Bandits.

Model Deployment Strategies

Strategies allow us to evaluate the ML model performances, capabilities and discover issues concerning the model. A key point to keep in mind is that the strategies usually depend on the task and resources in hand. Some of the strategies can be a great resource but computationally expensive while some can get the job done with ease. Let's discuss a few of them.

Shadow Deployment: To launch a model in shadow mode, you deploy the new, shadow model alongside the old, live model. The live model continues to handle all requests, but the shadow model also runs on some (or all) of the requests. This allows you to safely test the new model against real data while avoiding the risk of service disruptions. The shadow model handles all the requests just like the live model except it is not released to the public

This strategy allows us to evaluate the shadow model better by testing it on real-world data while not interrupting the services offered by the live model.

In shadow evaluation, the request is sent to both the models running parallel to each other using two API endpoints. During the inference, predictions from both the models are computed and stored, but only the prediction from the live model is used in the application which is returned to the users.

The predicted values from both the live and shadow model are compared against the ground truth. Once the results are in hand, data scientists can decide whether to deploy the shadow model globally into production or not.

But one can also use champion/challenger framework in a manner where multiple shadow models are tested and compared with the existing model. Essentially the model with the best accuracy or Key Performance Index (KPI) is selected and deployed [1-8].

Pros

- Model evaluation is efficient since both the models are running parallelly there is no impact on traffic.
- No overloading irrespective of the traffic.
- You can monitor the shadow model which allows you to check the stability and perfor- mance; this reduces risk.

Cons

- Expensive because of the resources required to support the shadow model.
- Shadow deployment can be tedious, especially if you are concerned about different as- pects of model performance like metrics comparison, latency, load testing, et cetera.
- Provides no user response data.

A/B Testing Model Deployment Strategy: A/B testing is a data-based strategy method. It is used to evaluate two models namely A and B, to assess which one performs better in a controlled environment. It is primarily used in e-commerce websites and social media platforms. With A/B testing the data scientists can evaluate and choose the best design for the website based on the data received from the users.

The two models differ slightly in terms of features and they cater to different sets of users. Based on the interaction and data received from the users such as feedback, data scientists choose one of the models that can be deployed globally into production.

Methodology
In A/B the two models are set up parallelly with different features. The aim is to increase the conversion rate of a given model. In order to do that data scientist sets up a hypothesis. A hypothesis is an assumption based on an abstract intuition of the data. This assumption is proposed through an experiment, if the assumption passes the test it is accepted as fact and the model is accepted, otherwise, it's rejected.

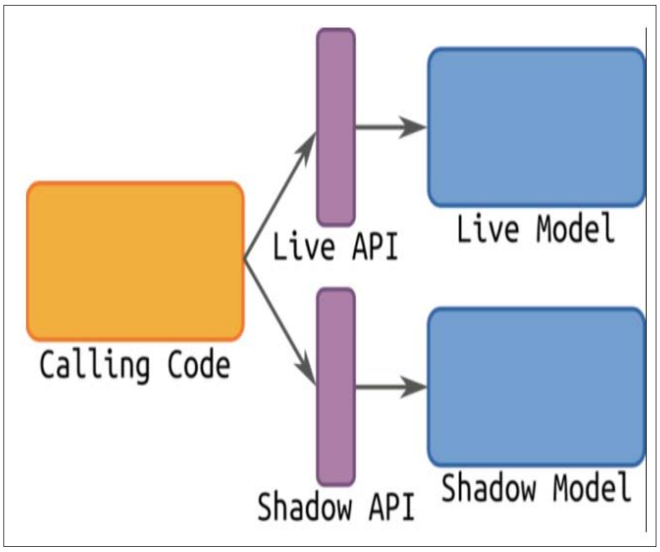

Figure 3: Shadow Deployment

Hypothesis Testing in A/B Testing there are Two types of Hypothesis

- Null Hypothesis states that the phenomenon occurring in the model is purely out of chance and not because of a certain feature.
- Alternate Hypothesis challenges the null hypothesis by stating that the phenomenon occurring in the model is because of a certain feature.

In hypothesis testing, the aim is to reject the null hypothesis by setting up experiments like the A/B testing and exposing the new model with a certain feature to a few users. The new model essentially is designed on an alternate hypothesis. If the alternate hypothesis is accepted and the null hypothesis is rejected then that feature is added and the new model is deployed globally.

Multi Armed Bandit: Multi-Armed Bandit or MAB is an advanced version of A/B testing. It is also inspired by reinforcement learning, and the idea is to explore and exploit the environment that maximizes the reward function.

MAB leverages machine learning to explore and exploit the data received to optimize the key performance index (KPI). The advantage of using this technique is that the user traffic is diverted according to the KPI of two or more models. The model which yields the best KPI is deployed globally.

Methodology MAB heavily depends on two concepts: exploration and exploitation.

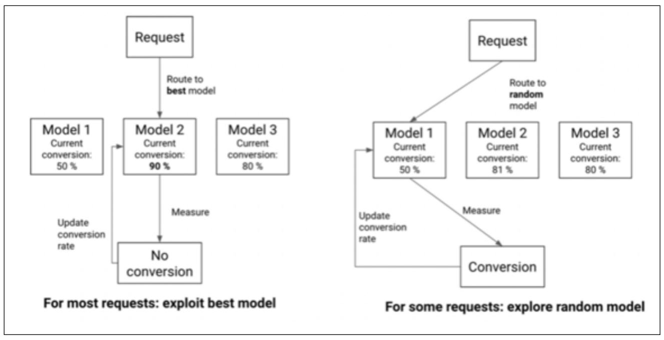
Request
Route to best model
Model 1 Current conversion: 50 %
Model 2 Current conversion: 90 %
Model 3 Current conversion: 80 %
Update conversion rate
Measure
No conversion
For most requests: exploit best model
Request
Route to random model
Model 1 Current conversion: 50 %
Model 2 Current conversion: 81 %
Model 3 Current conversion: 80 %
Update conversion rate
Measure
Conversion
For some requests: explore random model

Figure 4: MAB Deployment

Exploration: It is a concept where the model explores the statistically significant results, as what we saw in A/B testing. The prime focus of A/B testing is to find or discover conversion rates of the two models.

Exploitation: It is a concept where the algorithm uses a greedy approach to maximize conversion rates using the information it gained during exploring.

MAB is very flexible compared to the A/B testing. It can work with more than two models at a given time, this increases the rate of conversion. The algorithm continuously logs the KPI score of each model based on the success with respect to the route from which the request was made. This allows the algorithm to update its score of which is best.

Blue-Green Deployment Strategy: Blue-green deployment strategies involve two production environments instead of just models. The blue environment consists of the live model whereas the green environment consists of the new version of the model. The green environment is set as a staging environment i.e. an exact replica of a live environment but with new features. Let us briefly understand the methodology.

Methodology in Blue-green deployment, the two identical environments consist of the same database, containers, virtual machines, same configuration et cetera. Keep in mind that setting up an environment can be expensive so usually, some components like a database are shared between the two.

The Blue environment which contains the original model is live and keeps servicing requests while the Green environment acts as a staging environment for a new version of the model. It is subjected to deployment and final stages of testing against the real data to ensure that it performs well and is ready to deploy to production. Once the testing is successfully completed ensuring that all the bugs and issues are rectified the new model is made live.

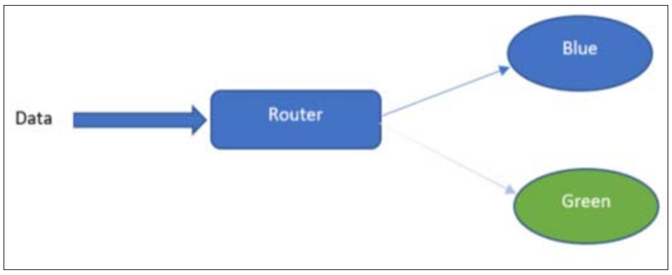

Figure 5: Blue-Green Deployment

Once this model is made live, the traffic is diverted from the blue environment to the green environment. In most cases, the blue environment serves as a backup, in case something goes wrong the request can be rerouted to the blue model.

Canary Deployment Strategy: The canary deployment aims to deploy the new version of the model by gradually increasing the number of users. Unlike the previous strategies that we've seen where the new model is either hidden from the public or a small control group is set up, the canary deployment strategy uses the real users to test the new model. As a result, bugs and issues can be detected before the model is deployed globally for all the users.

Methodology Similar to other deployment strategies in canary deployment, the new model is tested alongside the current live model but here the new model is tested on a few users to check its reliability, errors, performance et cetera. The number of users can be increased or decreased based on the testing requirements. If the model is successful in the testing phase then the model can be rolled out and if it is not then it can be rolled back with no downtime but only a number of users will be exposed to the new model.

Canary deployment strategy can be broken down into three steps:
- Design a new model and route a small sample of users' requests to the new model.
- Check for bugs, efficiency, reports, and issues in the new model, if found then perform a rollback.
- Repeat steps one and two until all errors and issues are resolved, before routing all traffic to the new model.

Conclusion
Deployment strategies often help data scientists to figure out how their model is performing in a given situation. A good strategy depends upon the type of product and users it aims to target.

There are various different scenarios which needs different deployment strategies that can help the overall product to iterate, test, build and deploy faster as a ecosystem. These should be highly modular to make the model lifecycyle reproducible.

References

1. https://alexgude.com/blog/machine-learning-deployment-shadow-mode/.
2. https://christophergs.com/machine.
3. Thomas H (2018) Davenport and Rajeev Ronanki. 2018. Artificial intelligence for the real world. Harv Bus Rev 96: 108-116.
4. Royal Society (Great Britain) (2017) Machine Learning: The Power and Promise of Computers that Learn by Example: An Introduction. Royal Society.
5. https://www.cloudbolt.io/blog/what-is-best-kubernetes-deployment-strategy/.
6. Michal Pěchouček, Vladimír Mařík (2008) Industrial deployment of multi-agent technolo- gies: Review and selected case studies. Auton. Agents and Multi-agent Syst 17: 397-431.
7. Kyle Wiggers (2019) IDC: For 1 in 4 companies, half of all AI projects fail https://venturebeat.com/2019/07/08/idc-for-1-in-4-companies-half-of-all-ai-projects-fail/.
8. Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, et al. (2019) Software engineering for machine learning: A case study. In Proceedings of the IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP'19) IEEE 291-300.