upGrad

INTRODUCTION TO MLOps

What do you mean by MLOps? Most of you might know that MLOps is the deployment of an ML model and creating an API endpoint from it. However, there is more to MLOps. We will learn about the same in this course.

Until now, you have seen how to create a model in a sandbox environment, that is your jupyter notebook, but is this how models are developed and deployed at an enterprise level? Let's try to understand all of these in this module. Many organisations are dipping their toes into machine learning and artificial intelligence (AI). Machine Learning Operations (MLOps) allows organisations to remove many of the obstacles they face in creating production-level ML solutions by providing a technical backbone for managing the ML life cycle.

The MLOps course is a starting point for anyone who wants to understand MLOps and ML engineering.

Elements that are covered in this module

1. **MLOps as a practice**: In the first session, you will understand how MLOps as a practice came into existence. In the previous module, you learnt about some DevOps practices; in this module, you will learn the similarities and differences between MLOps and DevOps. You will also look at the best practices of MLOps.
2. **ML system design process**: We will discuss MLOps practice in further detail and understand how to implement these practices/ principles to create MLOps systems.
3. **The building blocks of an MLOps stack**: Once we have discussed the flow of MLOps, we will understand what tools are available to help you across the process.

upGrad

Current State Of An ML Model In Production

The past decade has seen a lot of growth in the AI industry, as a lot of companies are trying to adopt AI to increase revenue. According to the MIT Sloan Management Review, 83% of CEOs report that artificial intelligence (AI) is a strategic priority. Gartner estimates that in 2022, $3.9 trillion in business value will be created by AI.

There is so much advancement happening in this industry, but many companies are still struggling to get the models into production. In 2019, VentureBeat said that 90% of the ML models could not be put into production, although that number decreased significantly in 2022. Read the article on why models were not able to make it into production here.

According to Algorithmia's 2020 report for which, Algorithmia had conversations with thousands of companies in various stages of machine learning maturity. It had polled nearly 750 business decision-makers across all industries from companies actively developing machine learning life cycles and came across the following insight around model deployment timeline.

More than 50% of companies take more than 30 days to put a model into production. The longer the deployment of the model takes, the lesser becomes the value of the models. This is because the data, patterns and the industry change so rapidly that the model that you built some months back may not be the best model for the current time. There are many other statistics in this report of Algorithmia. It is strongly recommended that you read this report.

There are numerous examples of machine learning models failing in production from companies such as Tesla, Facebook, Twitter, etc. The major issues here were

1. Model bias
2. Actual data being different from the historical data that we have trained our model on
3. No proper rollbacks
4. No stringent testing among others

You saw that many models failed even after they were put into production. Additionally, a lot of great companies failed, which shows that it is definitely not easy to maintain a model.

Can we afford to lose the reputation of our company over an AI model? Can we afford a loss of life over an AI model? No. Hence, proper processes need to be in place before you move your AI model into production.

Looking at a lot of failures in putting models into production must be extremely demotivating, right? But that does not mean we discontinue putting models into production. It means we should use a well-defined and accessible way to put them into production, which is MLOps.

upGrad

Origin Of MLOps

Google released a prominent paper called *Hidden Technical Debt in Machine Learning Systems in NIPS* 2015. There, Google applied the technical debt framework to machine learning systems and showed that they had designed a way to ensure **technical debt**, which we will discuss in detail later in this segment.

MLOps has its origins in DevOps, with its primary goal of bridging the gap between development and operations. But when it comes to machine learning, we try to extend these DevOps principles to fit the requirements of machine learning system development. MLOps has its root in the paper mentioned above. What does this paper say, and what is technical debt?

Technical debt occurs when you take shortcuts in the design or development process to move fast.

'Technical debt' is a phrase originally coined by software developer Ward Cunningham. He did not realise at that time that he had coined a new buzzword in the software community.

It is analogous to financial debt. Let's envision a scenario in which you want to buy a house, but you do not have money for the entire payment. You take a loan and then repay it with interest. It does not mean that taking a loan is bad, as, at that moment, it was required for you to buy the house.

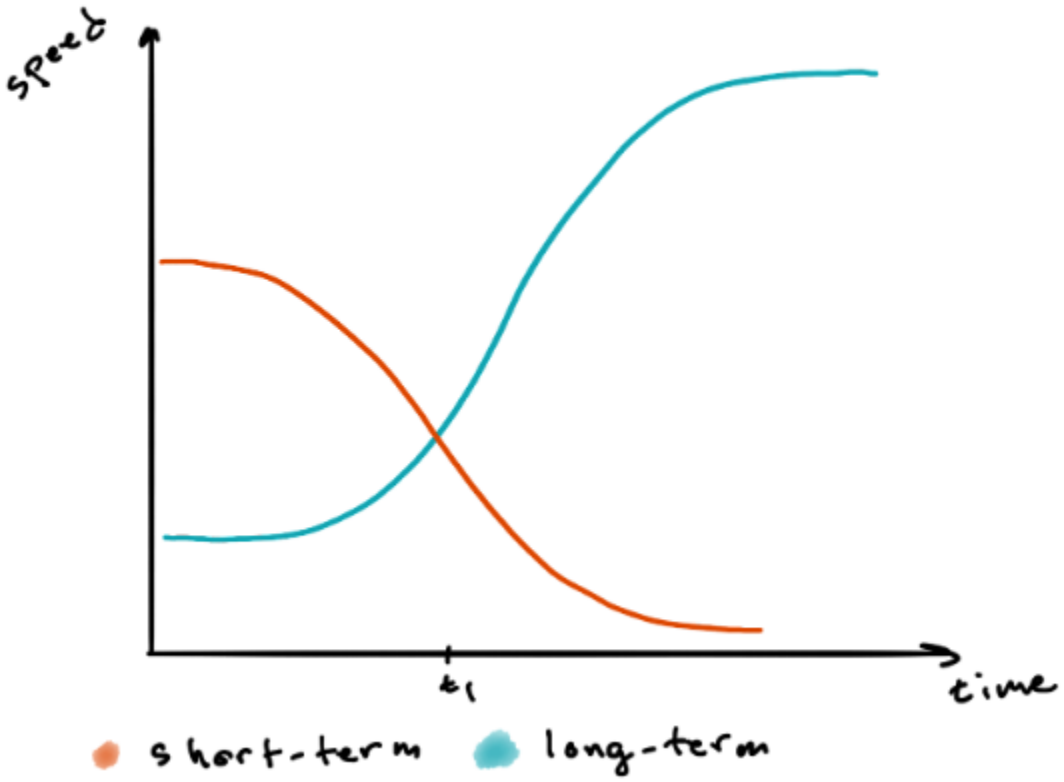
Similarly, businesses incur technical debt, as the product launch at that moment required them to. In software systems, technical debt can occur in scenarios where you wrote a code and put it into production without documentation or testing it.

The after-effects of not documenting the code will have consequences, as when the system fails, the other team who is debugging the code may not know where the error occurred and how to resolve it, as it was not documented.

Technical debt has to be repaid many times with interest. Therefore, it makes sense to pay off technical debt early on. So, one has to make a careful trade-off within a development team or a machine learning engineering team. Development team might want to get the best model into production and operations team wants to keep the uptime to maximum.

Technical debt is usually discussed in conjunction with short-term and long-term thinking. Utilising short-term thinking will lead to a scenario where you have a lot of technical debt that you will need to manage over time. When developing machine learning systems and ensuring they are production-grade, long-term thinking almost always makes sense. This will lead you to a scenario where you can release your model in a secure, quality-assured manner.

As you can see below, short-term thinking has an extremely high speed in the initial stages, but it drastically slows down in the long run, as the technical debt is very high. In the case of long-term thinking, although the initial stages take a lot of time, it is very fast later on.



Many times, when short-term thinking is implemented, the model usually does not make it into production. Even if it does, you will end up in a scenario where you are just solving some problem or the other related to the training or engineering aspect of the system.

The long-term approach is the scenario in which you are thinking not only about getting the project to launch but also about sustainability. If you have a long-term thinking approach in place, it might look that the progress initially is slower, but it gets a lot easier with time.

Let's take the example of building models in your jupyter notebook. You write the code in an unstructured manner and all the cells of preprocessing or modeling are spread across the notebook. You may get the accuracy that you need for your model to be a valid model. You then

get the results, and present the report to the business stakeholders. They are happy with it and ask you to move the model into production.

Let's take a scenario in which you have new data coming in, every hour. It will be very difficult to run your unstructured notebook every hour. You need to understand that not every cell needs to be executed in the notebook to make predictions. For example, cells that have df.head(), df.tail(), etc., are not required to be executed for the new data you are receiving. Hence, you need to identify the functions that are required to be executed in proper order for making inference. And then convert them into structured .py files as you had seen in the previous module. These files are then run in a scheduled manner as a part of an automated pipeline.

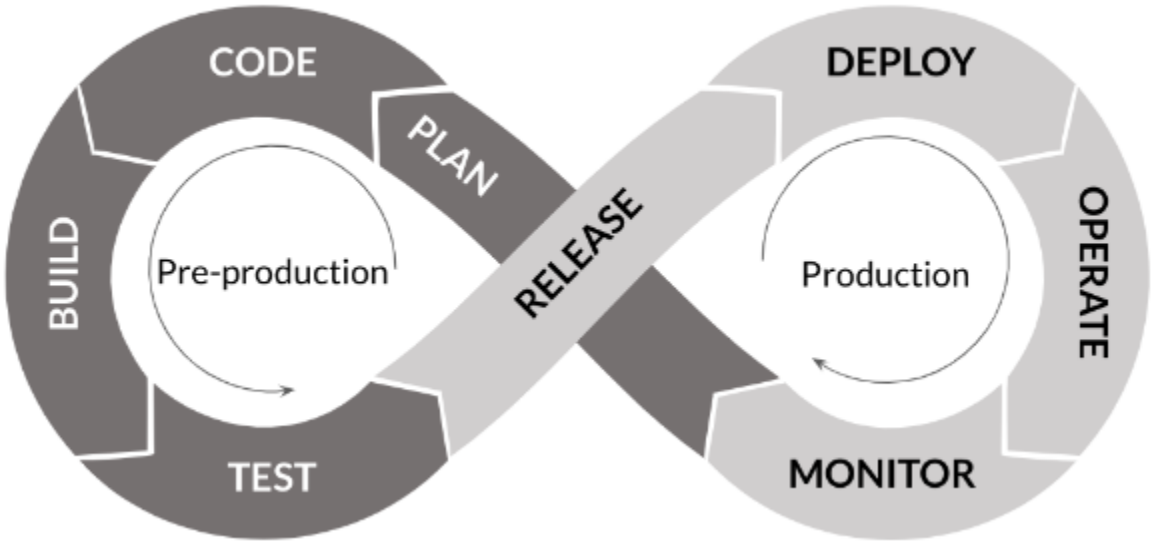
There can also be a scenario where you have not documented the code properly, and there is a problem a month later when the model runs on live data. The operations team that is debugging the code will not be able to understand what is going wrong in the code.

According to the paper Hidden Technical Debt in Machine Learning Systems, "Developing and deploying machine learning systems is relatively fast and cheap, but maintaining them is costly." In general, within the machine learning community, it is believed that developing and deploying machine learning systems is extremely easy and fast. Therefore, you end up skipping a lot of stages that are required to maintain your machine learning system. These issues are something that we aim to tackle with MLOps principles.

In the beginning, the machine learning community tried to put models into production using DevOps principles but failed. This was because Machine Learning had some inherent differences from traditional software, which is why the term 'MLOps' came into existence.

upGrad

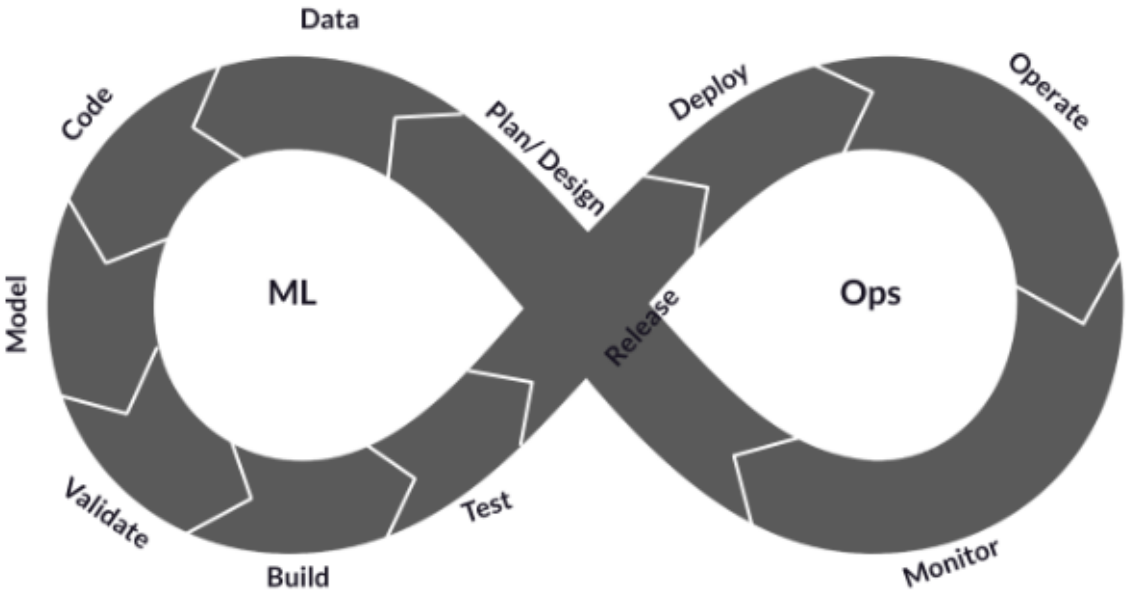
Machine Learning Software Vs Traditional Software



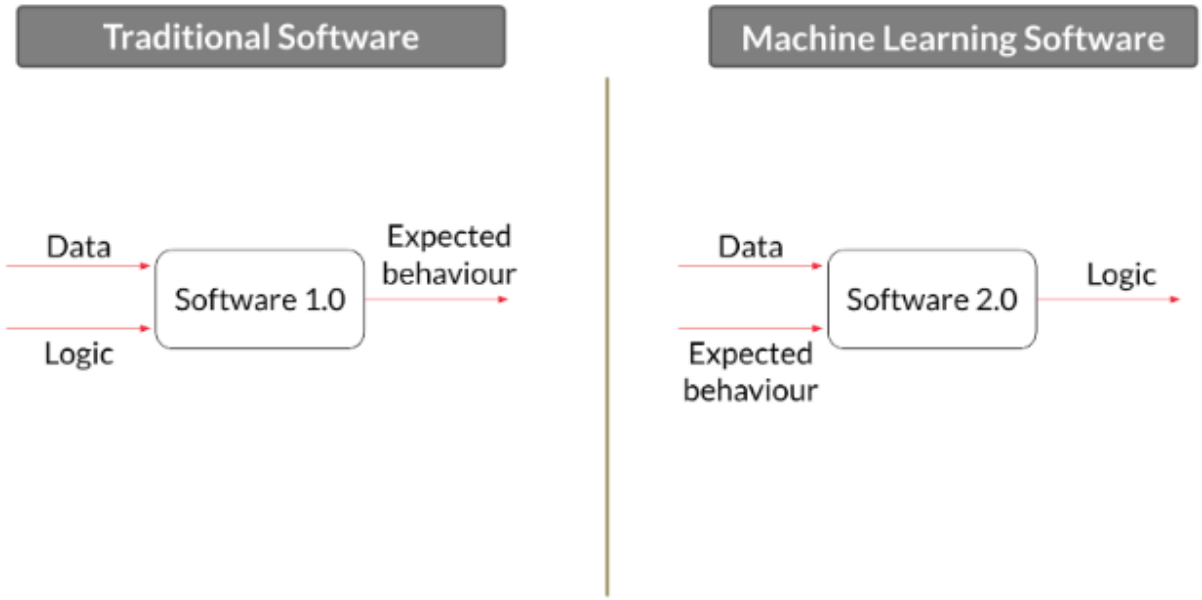
The major difference is the addition of the model and the data phase to the DevOps cycle. Simply put:

MLOps = ModelOps + DataOps + DevOps

The concept of development operations (DevOps) is not new. It has been used for decades to deploy software applications, and the deployment of ML applications has much to gain from it. However, strong DevOps practices and understanding of tools are insufficient for deploying ML use cases. This is because ML applications rely on models which require continuous training to keep up to speed with the continuously changing data patterns.It is not that the data and the model phases are simply added as the initial steps; their effects cascade down the line as well. For example, in testing, we need to test the code for the model and data processing steps as well. Similarly, in monitoring, we need to monitor the model and the data as well.



More fundamentally, traditional software and machine learning software have inherent differences, as shown in the diagram below:



Let's summarise the differences between traditional and machine learning softwares:

1. Traditional softwares are deterministic, as logic is provided to them, whereas in machine learning softwares are very probabilistic, as we derive logic as an output.
2. In traditional software, there are only code-level tests, whereas in machine learning software, we have to test the code for data processing and models as well.
3. In traditional softwares, monitoring helps track the availability and performance of the compute resources, but in machine learning softwares, we also have to monitor the data patterns and model results. Take the example of Meta labelling images of black men as primates. Although this service was running and labelling these images, the actual model was completely flawed.

In summary, machine learning system development inherits all of the complexities that accompany the delivery and maintenance of traditional softwares. In addition to this, you have other complexities that are unique to machine learning systems, mainly related to the dependencies on data and the nature of the algorithms themselves.

upGrad

MLOps Best Practices

The following are some of the best practices of MLOps:

Reproducibility and reusability: This will help the development team to reproduce the experiments.

- **Automating as much as possible**: There can be different types of automation, such as automated testing, automated training, automated pipelines, etc.
- **Testing**: Testing is necessary for data, models and code used for integrating all the elements.
- **Adopting monitoring and alerting**: Monitoring is very important in the machine learning context because of the silent failures that happened in the case of Twitterbot and Zillow.
- **Explainability over performance**: Being able to explain why you are making a certain prediction is often more valuable to the business than the prediction itself. So, in many contexts, try to stick to explainable modelling approaches.
- **Doing risk assessments**: Before pushing your machine learning models into production, make sure that you spend some time doing risk assessments so that your model is robust and does not fail in production. Ensure you have a plan of action because you might end up in a scenario that will deeply hurt your brand or even cause the failure of your startup.
- **Automated rollbacks**: Automated rollbacks are necessary in case of any disaster

Additional Links:

Here are some popular links to MLOps best practices and a few references that encapsulate a lot of MLOps principles and good practices to keep in mind:

1. Engineering best practices for Machine Learning by SE4ML
2. Rules of Machine Learning: Best Practices for ML Engineering by Martin Zinkevich

upGrad

DISCLAIMER 1: This learning module may contain images, both still and moving, including photographs, advertisements, third-party trademarks, video clips etc. that may constitute copyrighted material procured from open-source, public domain, social media, or other print or digital publications, with the sole and bonafide intention of imparting education, disseminating information and illustrating an academic concept or an issue. We believe that this constitutes a 'fair dealing' exception under Section 52 of the Indian Copyright Act, 1957.

DISCLAIMER 2: Throughout this course, you will see examples of different brands and institutes. These have been chosen from specific cases in a way that would aid in demonstrating concepts. These do not reflect the current performance of any company. Any opinion expressed is strictly for educational purposes.