Pluto7

MACHINE LEARNING OPS

CONTENT

INTRODUCTION

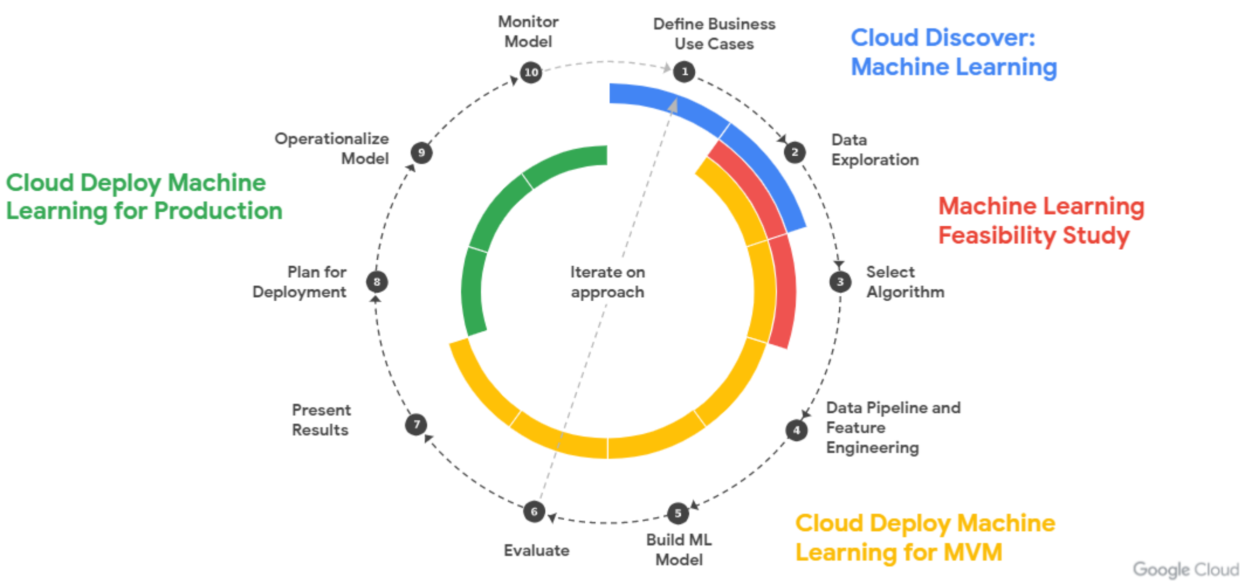
There is no doubt regarding the fact that Machine Learning is changing the world. Hailed as “next electricity” by Andrew Ng, it comes with a lot of promises. Despite these promises, the internet is overflooded with statistics and anecdotes about Machine Learning and Data Science products failing to make it to production. One such report, by VentureBeat, claims the number to be 87%. This paper aims to look into the problem of why only one in ten machine learning projects make it to production and how MLOps can help organisations to effectively manage machine learning systems in production.

THE NEED FOR ML OPS

A brief history Software Development Lifecyle (SDLC):
In the last couple of decades, Agile methodologies have revolutionized software development. Rather than having a one-shot launch (waterfall model way), Agile methodologies introduced an incremental and iterative approach towards innovation. While “Pure Agile” terminated after deployment, a product or solution required constant monitoring, so that response to changes can be made as quickly as possible.

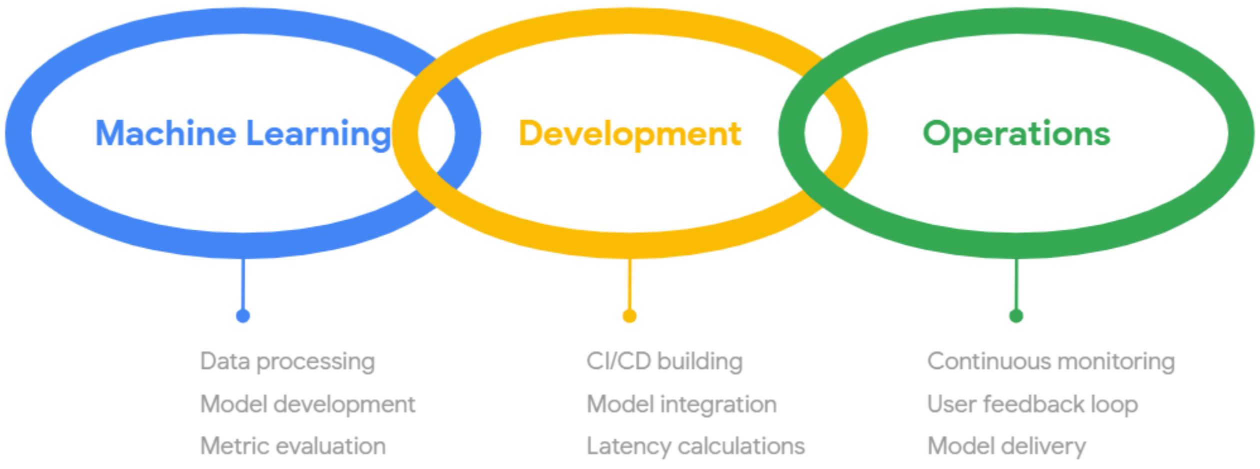
This required development and operations to become “one”, and hence, DevOps was born. Though a relatively recent phenomenon, DevOps is now a standard way of managing software lifecycles.So, the question arises, if DevOps is the standard across the industry, why is it not suitable for Machine Learning systems? The answer is fairly straightforward. Machine Learning applications contain “data”, which is generated by an infinite entropy source, the “real world”.

DevOps does a great job of managing the code, but managing a large number of data sources and models (weights and biases) is altogether a different ball game. So, as in-production solutions called for continuous monitoring, which led SDLC to evolve from Agile to DevOps, now, as our solutions are becoming more and more data-savvy, it is time for us to make progress in the SDLC ladder. ML modeling and productionalizing has adopted a similar lifecycle as shown below.

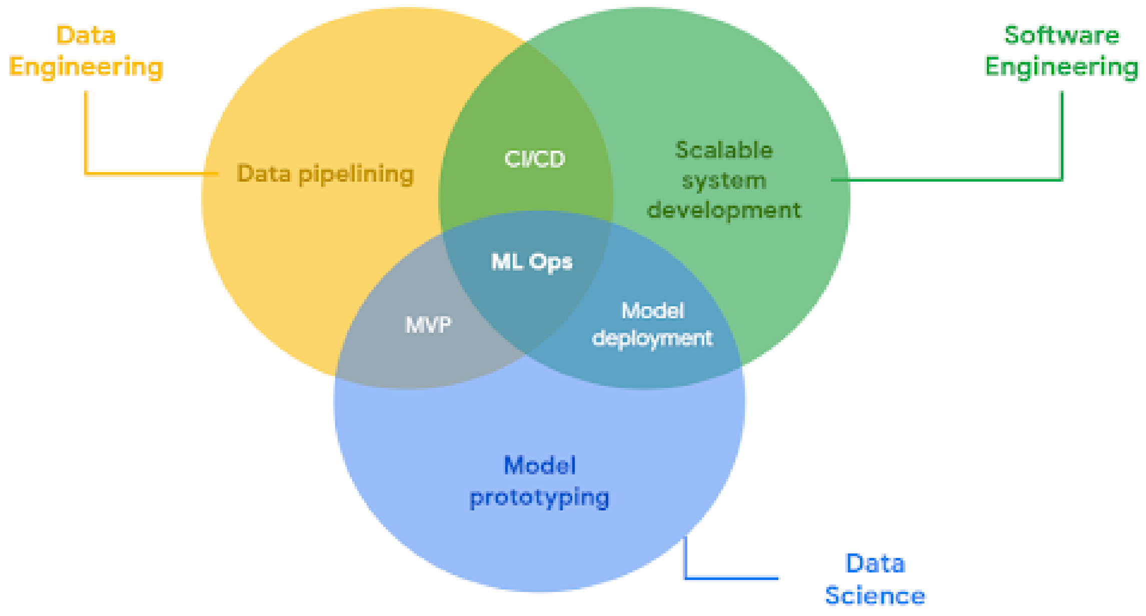

Monitor Model
Define Business Use Cases
Cloud Discover: Machine Learning
10
1
Operationalize Model
9
Data Exploration
2
Cloud Deploy Machine Learning for Production
Machine Learning Feasibility Study
Plan for Deployment
8
Iterate on approach
3
Select Algorithm
Present Results
7
4
Data Pipeline and Feature Engineering
6
5
Evaluate
Build ML Model
Cloud Deploy Machine Learning for MVM
Google Cloud

The problem with traditional approaches:
The problem with adopting traditional methodologies for Machine Learning is that it is very difficult to define “incremental progress” in machine learning, which is at the core of Agile. A machine learning model usually contains data streams and a machine learning model (which at its core is nothing but numerous matrices of numbers).

These are accompanied by pre-processing and post-processing pipelines, making data ready for machine learning models and human consumption respectively. Deploying any one part without the other, makes no impact. One part of this holistic system doesn’t enhance a solution or a product, unless or until all of them act in synergy.



Before the advent of machine learning, data was part of certain applications and there were certain methodologies that were used to manage such projects. The Knowledge Discovery in Databases (KDD) was used in data mining applications, but it is rigid, making it difficult to adopt it for continuously evolving training and prediction cycles.



Redefining Agile in the context of ML:
There are a few challenges that we need to take into consideration while deploying a model. These are:-

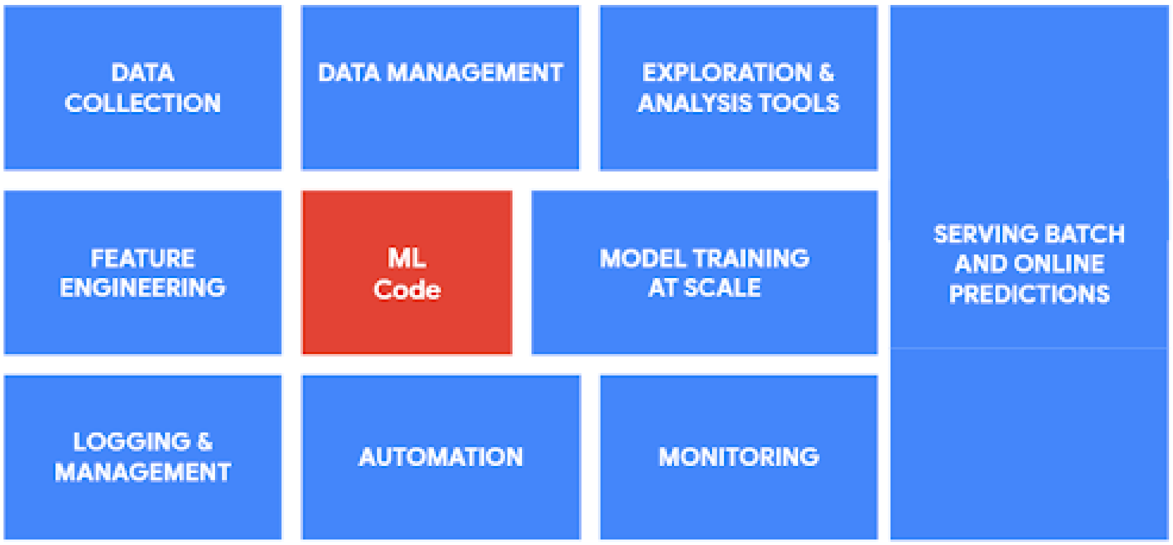
- The environment in which the model is deployed, is itself affected by the model. This affects the assumptions that we took into consideration while building that particular model. This is especially the case in recommendation engines and personalisation engines. Because the ranking of products and ads are the result of model predictions, it invariably starts affecting the user inputs. This feedback loop introduces the selection bias, which wasn't present earlier when the model was being built.
- No matter how good the model is, there is always a downfall in the prediction power. This calls for new retraining at continuous intervals.

To meet these challenges of putting machine learning models in production, a SDLC process needs traditional elements like Continuous Integration and Continuous Development but we need constant monitoring and Continuous Training elements also, which trains the model at necessary time intervals. The MLOps bring the necessary philosophies, processes and tools under one hood and streamline the process of putting machine learning models in production.

SOLUTION

The MLOps consists of three primary stages, Continuous Integration, Continuous Development and Continuous Training. The setup for MLOps must include the following:-

- Easy experimentation - A machine learning engineer should be able to iteratively try out different sets of models, different feature engineering techniques and come out with a good machine learning model, which can be readily deployed.
- Feature Store - A feature store makes sure that the consistent features are being served in the training and inference pipelines.
- Continuous Training - validating the currently deployed model's performance and triggering the training job if required.
- Continuous Integration - The continuous integration can help us to test, integrate and build the changes to the machine learning pipeline.
- Continuous Delivery - The developed model could be easily deployed for serving and can be integrated to the existing application easily.
- Orchestration Engine - The orchestration engine is an automation engine that triggers training with the predefined schedule, upscale and downscale the system depending upon the request traffic and other configuration parameters.



Pluto7

Experimentation
The experimentation is at the centre of any machine learning flow. Though the jupyter notebooks are the go to tools of the data scientists, they are at a considerable disadvantage when it comes to collaboration. The AI Platform Notebooks are a considerable improvement over the traditional on prem notebooks. They are basically managed Jupyter Notebooks instances on the Google Cloud Platform. It comes with various configurations and with libraries pre-installed. So, the developers need not to configure the environment on their own. Other advantages offered by the Jupyter Notebooks are:-

- Can be vertically scaled up or down according to the load
- Natively integrated with other services like BigQuery, Cloud Storage
- Google Cloud Source Repository support

Feature Store
One of the ways we can avoid training-serving skew is by calculating our statistical summaries on the training dataset only and using that same summaries with the testset. The problem that arises here is how to store these calculations. This is the place where Feature Store comes handy. The Feature Store is a recent development which provides us with a centralized storage layer where we can read and write features, for training and serving purposes. The feature store provides with following benefits:-

- Reuse features, entities, sets and embeddings instead of running the job again and again.
- Can serve batch and near real-time inference.
- With every training session, we can update the features and serve them with every respective inference.

We can leverage Google Cloud Platform’s BigQuery to create our own custom feature store. The BigQuery provides following benefits, which makes it a go to tool for this purpose:-

- BigQuery is Severless. So, no hassle to manage the infrastructure.
- BigQuery allows SQL to query the table
- Scales to petabyte scale

Continuous Training
The Continuous training is performed by automating the model development, where the model is retrained based on various triggers, which are marked by pipeline validation. Pipeline Validation is a process of making sure whether the deployed model is relevant or not. The machine learning pipeline expects a model to be tested and trained

automatically, without any manual interference. A machine learning model is made up of two components, data and the machine learning algorithm, so, we need to test both of these inputs, in order to arrive at a decision.

- **Data Validation** - because the source of data is the real world, it is very susceptible to the changes. These changes could be covariate shifts, which is a change in the underlying probability distribution of the data. This can result in training serving skews, massively impacting the performance of the data. In the case of training serving skews, we need to trigger the model training. The other problem could be unavailability of the feature due to various reasons. This requires the development team to brainstorm why it happened and how to overcome this situation.
- **Model Validation** - the model validation is performed to check the performance of the model on different subsets of data and evaluating its prediction consistency. This can be done by capturing performance metrics across different subsets of data.

Continuous Integration

The continuous integration can help us to test, integrate and build the changes to the machine learning pipeline as soon as the code is committed to the Cloud Source Repository. We can leverage GCP's Cloud Build for creating such pipelines. Though the model itself can't be tested on it's own, due to varying performance metrics, the feature engineering logic and post-processing code can be unit tested. For instance, we are only accepting a categorical input for the columns that are to be Label Encoded or continuous input for the column that is to be bucketised.

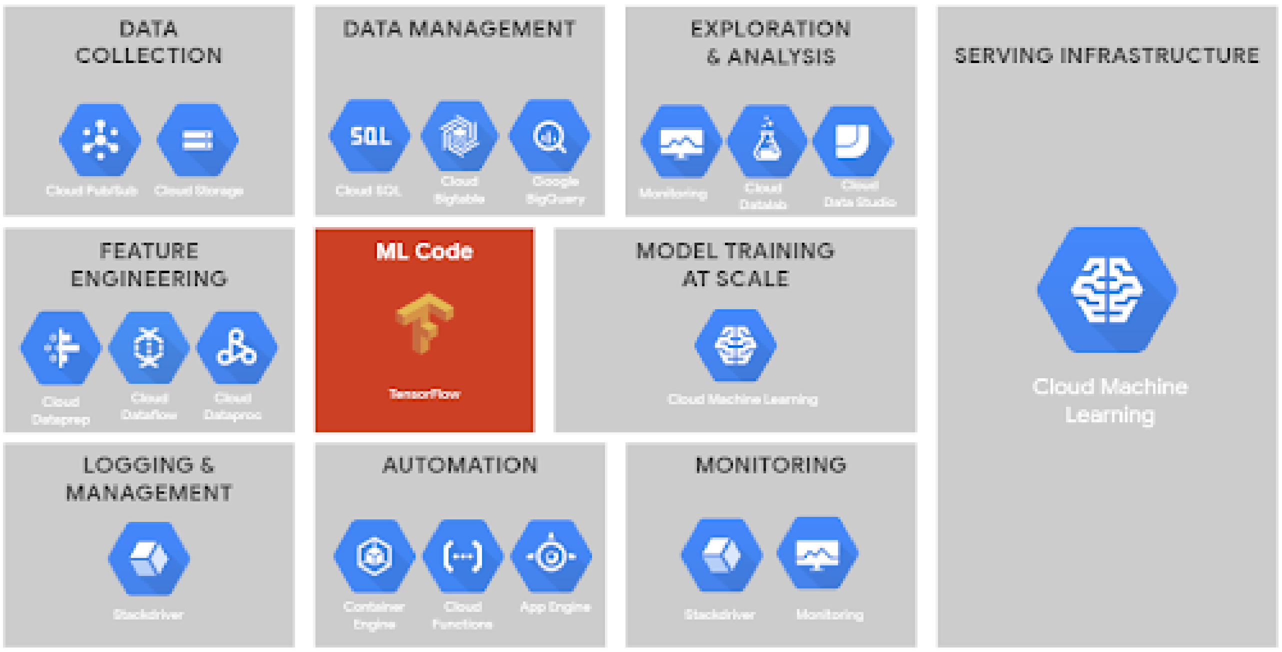
Continuous Delivery

The continuous delivery pipeline continuously delivers the new changes made to the machine learning pipeline to the serving environment. A good continuous delivery pipeline has following characteristics:-

- A CD pipeline should check all the dependencies that are required for seamless execution are already installed in the serving environment.
- It should be automated to the maximum extent and require as little manual interaction as possible. An example of this could be automatic deployment as soon as the code is merged to the master branch. It follows the rule that anything in the master branch deployment-ready, so an extensive code review definitely helps.
- Able to provide the inference via a callable API. It makes model testing quite easy and approachable. We can configure API input as such it doesn't hinder the serving pipeline.

Orchestration

The orchestration engine is an automation engine that triggers training with the predefined schedule, upscale and downscale the system depending upon the request traffic and other configuration parameters. For this component of the MLOps, we can use Cloud Composer which is a managed Apache Airflow service.A benefit of using composer is that it allows us to build pipelines that can be multi or hybrid cloud. This covers all the aspects that are involved in the MLOps. The following table illustrates the complete solution of MLOps on GCP. We can use AutoML as our model development tool, for it automates the model training and deployment. GCP has many AutoML offerings, depending upon whether we are dealing with unstructured and structured data.



Maturity Curve with GCP Adoption

Maturity	Characteristic	Leverage GCP Capability
No ML Ops	Early stage ML model building and deployment by engineers with no clearly agreed terminologies and support plan. Manual ML model building, testing, deployment and maintenance Model performances are tracked manually.	Auto ML GCMLE
Early Stage ML Ops	Model build and testing	Kubeflow
	automated by production management of models manual	Stackdriver Data Studio Audit Dashboards
Advanced ML Ops	Complete automation of ML model, build, train, deploy and manage	Ai Platform capabilities of model version management, Kubernetes, Istio on Anthos

CONCLUSION

Though MLOps is in its nascent stage but it is evolving continuously. Google recently open sourced Kubeflow, which makes deployment of machine learning pipelines very easy and scalable. With multi framework support, it allows to compose, deploy and orchestrate machine learning pipelines. The efficiency of a MLOps setup can be measured by calculating how quickly a new approach can be tested at scale, because the ethos of software development is still "failing fast".

References:

1. The hidden technical debt of machine learning systems
2. MLOps: Continuous delivery and automation pipelines in machine learning
3. The architecture of MLOps using TFx, Kubeflow and Cloud Build
4. Knowledge Discovery in Databases