CLOUDERA

EBOOK

BLUEPRINT FOR CONTINUOUS OPERATIONS FOR PRODUCTION MACHINE LEARNING

By Jeff Fletcher and Ade Adewumni

Table of Contents

Introduction

More organisations are adopting machine learning (ML) and the market has matured enough to be able to offer more options and capabilities. However, in many cases the business value derived from their investment in ML has been inconsistent. This isn't because enterprise customers have been careless in their approach to machine learning. Many were careful to conduct 'science experiments' to identify which of their business challenges machine learning could be successfully applied to. The problem is that success in developing these proofs of concept (PoCs) does not necessarily translate into the successful 'operationalization' or deployment of machine learning. Further, this type of experimentation is not cheap. Outside of the so-called Big Tech firms[1] few organisations can sustain extensive experimentation without clear evidence that it will contribute to the bottom line. If none or very few of the successful PoCs make it to deployment, it's difficult for organizations to justify ongoing investment. This is true even if it's been demonstrated, through PoCs, that machine learning can be successfully applied to the underlying business challenge.

Enterprise customers are realising that in order to see a return on their investment in machine learning, these successful ML experiments need a formalised path to deployment and a structured approach to post-deployment maintenance. They are also aware of the importance of doing this in a way that minimizes costs and friction which make it harder to sustain the value of these ML applications in the medium to long term. This is where the methodology, Continuous Operations for Production Machine Learning (COPML) can help. COPML is a comprehensive approach to model development and operation that facilitates the creation of machine learning applications which deliver continuous business value for as long as possible and can be smoothly retired once this ceases to be the case.

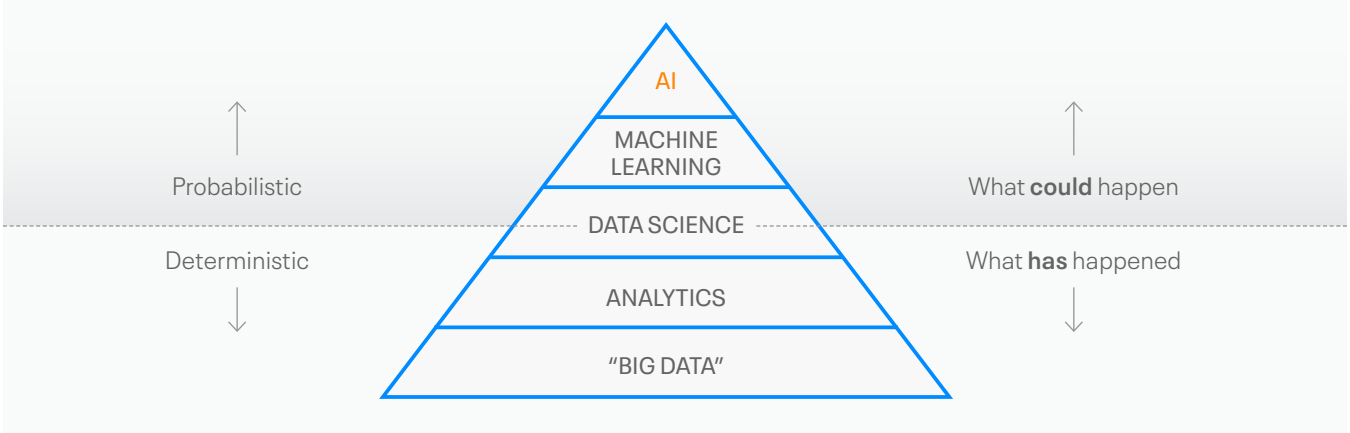
The scope of COPML isn't limited to model artifacts or the tools and systems that support their ongoing maintenance and ensure that they are smoothly served to downstream applications. It also supports the fostering of collaboration between the various teams involved in the machine learning workflow. This document provides an exploration of the COPML methodology, using two example use cases to further illustrate what its application could look like in practice.

1: Defining Production Machine Learning

Machine learning is already an overloaded term so perhaps a good starting point for this document would be to clarify what we mean by Production Machine Learning. In our view, it is the point at which the output of a machine learning model fits into the broader business environment which is where value is realised. The term comprises the various processes and practices that enable this to happen in a consistent and sustained manner.

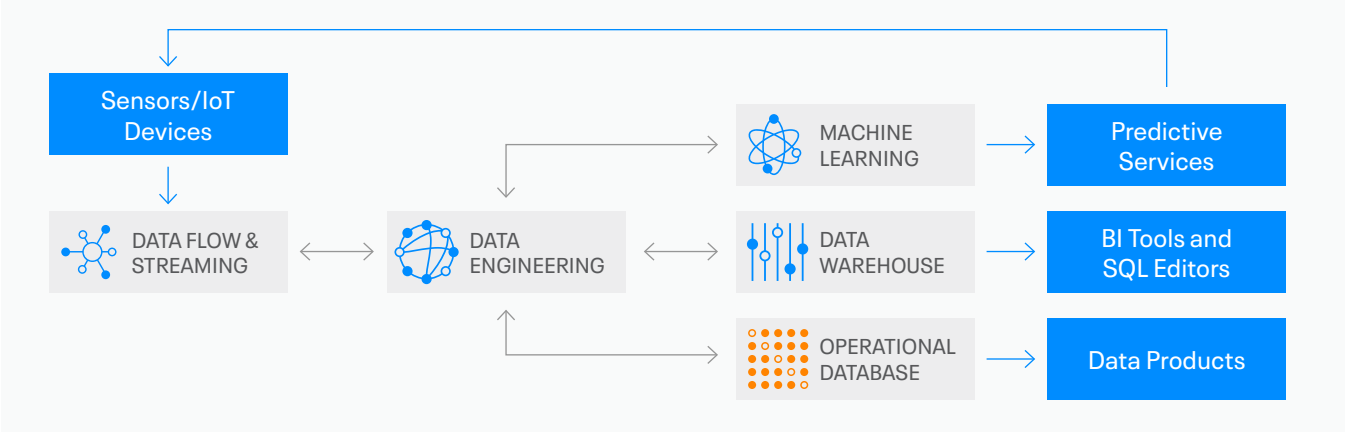
Data Science vs Machine Learning vs AI - get the right tools for the job

While machine learning is technically part of the data science discipline, in an enterprise context it's often treated as its own function. This is probably because of the type of implementation demands that machine learning makes on organizations relative to other forms of data science. These demands reflect the different purposes (and resulting value) of machine learning when compared with other data science sub-disciplines.



A simple way of thinking about the differences between analytics, data science and machine learning is to consider the different types of questions they allow the business to answer. Is the requirement to use existing data to understand what has already happened or to make a prediction about the future? Machine learning enables the latter while analytics supports the former. Predicting what could happen in an area which is of interest to the business, can be a very useful source of competitive advantage. However, realising this competitive advantage depends on organisations being able to incorporate these predictive insights into their business processes. Production machine learning is focused on implementing machine learning in a way that allows businesses to maximize the resulting benefits for as long as possible.

It's also important to understand that machine learning is not and should not be considered its own island, operating independently from the other data disciplines and with its own separate infrastructure. Most enterprise production machine learning projects will use and generate data that has utility to other parts of the business (and other data science disciplines!). It's important that production ML is implemented in a way that facilitates the ability to access, update and augment relevant data using the same platform the rest of the business uses.

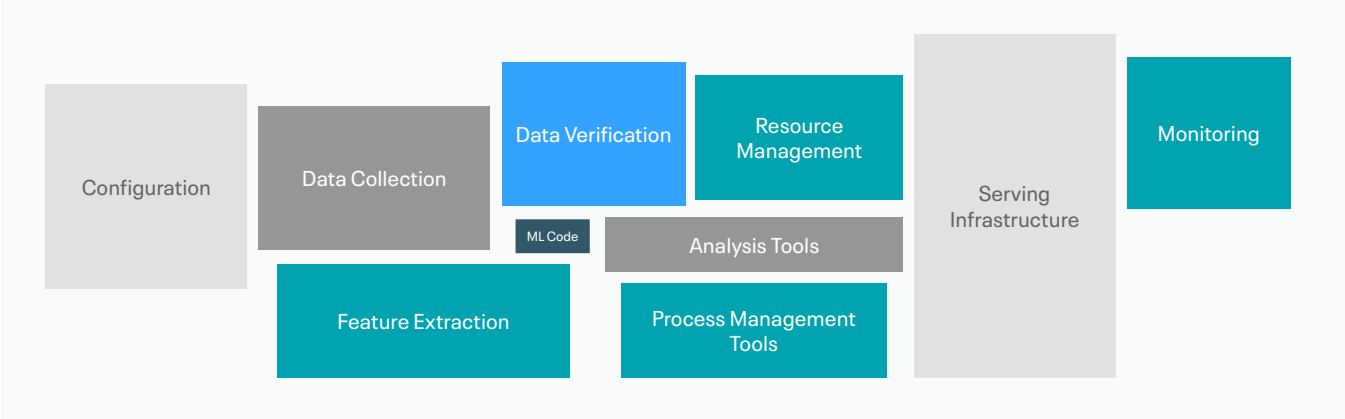


Having said that, there are some additional infrastructure and platform requirements that are specific to machine learning which need to be considered. Training complex neural networks may require GPUs or specific periodic, elastic ML infrastructure that would need the auto-scaling capabilities of public cloud. Ultimately there is a lot of overlap between the data, the various data teams and the infrastructure that they use. When looking at the infrastructure requirements presented later in this document, it's worth first knowing what is available and applicable before considering bringing in something new.

Machine Learning in Production is Hard

There is plenty of research focused on the challenges of getting machine learning projects into production.

The diagram below is taken from one such piece of research, Hidden Technical Debt of Machine Learning systems.

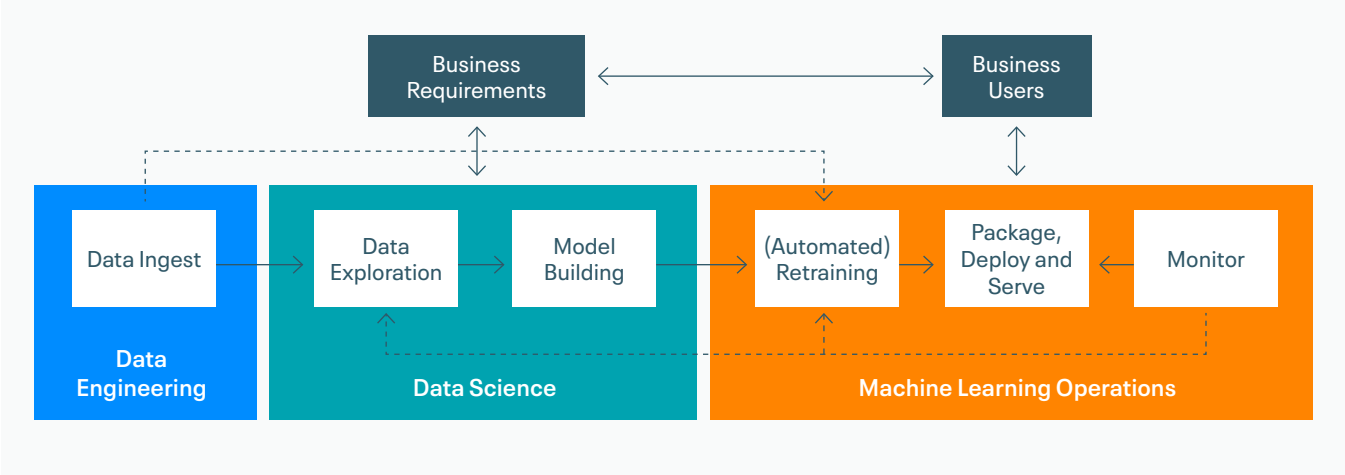


One of the key take-aways from the paper is that the development of machine learning code represents a very small part of the much bigger and complex production machine learning ecosystem. Enterprise data scientists can probably find sample models for some of the more common machine learning use cases via an internet search. These sample models will probably come with clean data (likely in the form of a zipped CSV file) and a Jupyter Notebook that implements all the code. This might provide a good basis for developing a working model. Alas, this will not suffice for real world production machine learning. All of the other parts of the data ecosystem need to be running optimally for the ML code in the middle to be useful. It is the integration of that machine learning code with the rest of the ecosystem (and the teams that run it!) that is the hard part.

This reality is often obscured by the hype surrounding data science which inflates the importance of model development relative to other parts of the ecosystem. As an example, consider a financial services institution in which the operational database breaks down, limiting the availability of transaction data to the downstream applications. This is a significantly more important issue than a break-down of an ML application which predicts which customers might churn. While model development is, understandably, very important to data science practitioners, it often does not enjoy the same degree of importance within the wider business. As a result, data scientists' needs are not necessarily prioritized when it comes to technical decisions that may ultimately impact their work. Consequently, while they may be able to operate independently when it comes to developing a ML Proof of Concept (PoC), they may struggle to take things further. This inability to integrate the model output into the rest of the ecosystem will ultimately result in the project's failure.

2: Production Machine Learning Workflows

This section takes a look at the workflow that is most often applicable to enterprise machine learning projects. It is not the only implementable workflow, but it is the most common one we, as a customer facing field team, see.



Workflow Steps

The steps below follow the typical workflow for taking a machine learning project to production.

STEP 1: CLARIFY BUSINESS REQUIREMENTS

Everything starts with a business requirement; we need to be able to predict X in order to meet business requirement Y. The source of this request can be from anywhere within the business, and it will likely come from a team or department with limited data science and machine learning capabilities. The request may not be valid or relevant to implementing a machine learning project though. To make sense as a machine learning project the request should include a requirement to predict something that is currently unknown and at least one measurable metric that can gauge the success of this project.

STEP 2: ASSESS AVAILABLE DATA

Once the data science team has understood and validated the project, they need to determine what data will be needed for the project. This means looking internally at what data is available, checking if it is accessible, and what gaps there might be. If additional data is needed, the data science team should work out if this data can be obtained internally or needs to be acquired from an external 3rd party.

STEP 3: DEVELOP THE DATA SCIENCE PLAN

The next step is exploratory data analysis. This is to understand the shape and structure of the data. This is also where the data science team will come up with a plan for developing a model that solves the business requirements. This is also where the initial model building, testing, experimentation and optimization comes in. The end result should be a working prototype of the machine learning model that will be deployed in the next step.

STEP 4: MODEL DEPLOYMENT

The model developed in the previous step needs to be deployed into production. This means it needs to be deployed in a way that allows its output to be used by the business (or more realistically, other internal applications) for the predictive requirement. It needs to be deployed in a way that matches the availability requirements expected by the teams and internal applications that will use it.

STEP 5: MODEL OPERATIONS

The final step is the automation and continuous monitoring of the model. This will include monitoring metrics and availability of models and other parts of the end to end implementation. This is also where tasks are triggered and run to update, train and redeploy models to make sure it continues working effectively over time. This step should also include assessing when a model is no longer needed by the business. A model could still be operating effectively but there is no longer need to use it within the business and it should be shut down to get back the resources.

Example Scenarios

The following examples showcase complete implementations of production machine learning projects. Both examples have been implemented within the Cloudera Data Platform using the Applied Machine Learning Prototypes (AMPs) feature in CML. They can serve as useful guides for how you might implement similar projects, using your own data.

Scenario 1: Churn Prediction

Github link: https://github.com/fastforwardlabs/COPML-AMP-1-telco-churn

Let's apply this workflow to a business challenge – predicting the likelihood that a customer will churn.

id	Probability	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
951	0.698	Male	No	No	No	22	Yes	Yes	Fiber	No	No
2999	0.638	Male	No	Yes	Yes	1	Yes	No	Fiber	No	No
5400	0.638	Femal	Yes	No	No	26	Yes	Yes	Fiber	No	Yes
6315	0.494	Femal	No	Yes	Yes	9	Yes	No	Fiber	No	No
5072	0.115	Male	No	Yes	No	67	Yes	Yes	Fiber	No	Yes
5611	0.109	Male	No	Yes	No	72	Yes	Yes	Fiber	No	Yes
5020	0.053	Femal	No	No	No	72	Yes	No	Fiber	Yes	Yes
5973	0.053	Male	No	No	No	15	Yes	No	No	No in	No in
6720	0.047	Male	No	Yes	Yes	15	Yes	No	No	No in	No in
2029	0.031	Male	No	Yes	Yes	41	Yes	No	DSL	Yes	Yes

Applying this workflow to the original telco churn prototype would yield a number of steps:

STEP 1: CLARIFY BUSINESS REQUIREMENTS

A fictitious telco business wants to predict which customers are likely to churn in order to be able to reduce the current churn rate (e.g. from 10% to 5%). In order to fulfil this objective the business needs to be able to predict the probability of any of its customers churning. Those deemed as 'high risk' can be entered into some sort of remediation process. For example, an offer of a free data or text package, or whatever the business has decided is the best course of action for retention.

STEP 2: ASSESS AVAILABLE DATA

The data science team assesses the customer-related data that's available in the organisation's data warehouse and any relevant data that's been made available from other sources and confirms it is accessible to the systems the data science teams will use for EDA and model training. In this particular case, the customer-related data includes demographics information, usage data, product mix, monthly charges, total charges etc. It is important that the available data includes both customers that have left (i.e. those that have churned) as well as existing customers who have not churned.

STEP 3: DEVELOP THE DATA SCIENCE PLAN

For this project, the plan is to build a binary classifier model that can classify a customer as a churn risk or not. This model can be applied to any existing customer. The data science team will create notebooks for conducting exploratory data analysis and for the machine learning model build.

STEP 4: MODEL DEPLOYMENT

The model needs to be deployed into the place where its output is available to the relevant teams so they can fulfil the business requirement. Our illustrative code builds a real time model that can make new churn predictions for customers on the fly. However, the more common scenario would be one implemented using a batch process. For example, a batch job might be run once a week, its output would update a table or a list that is accessed by the team that's tasked with implementing the customer retention strategy.

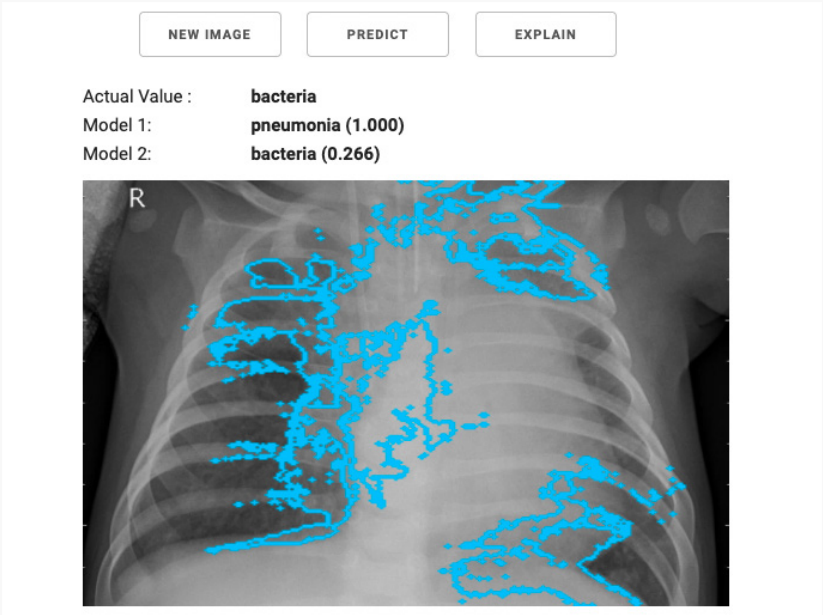
STEP 5: MODEL OPERATIONS

The model's performance needs to be checked periodically. A good way to do this is to examine a proportion of the customers that the model made predictions for and assess the accuracy of those predictions e.g. how many of those predicted to churn actually did? Alternatively, what are the precision and recall values of the model? If the model performance falls below an acceptable level then it will be necessary to retrain the model. The performance metrics in this scenario are relatively straightforward, but even one this simple can be hard to put into production within an enterprise context. For example, the choice about how often to assess the model's performance is very dependent on the business circumstances and the consequences of inaccurate predictions or concept drift. In the churn prediction scenario described here, while timing matters it's not the most important thing. Churn has business implications in the medium- to long-term but its short-term impact is limited. Therefore a delay of a few days for a performance assessment would probably be acceptable.

Scenario 2: Detecting Pneumonia in Chest X-Rays

Github Repo: https://github.com/fastforwardlabs/COPML-AMP-2-xray

This computer vision challenge calls for a different approach to implementing a machine learning model. The requirements are very different from the Customer Churn scenario above. For one thing, given the health context, the risks associated with a poor or inappropriate implementation is far greater.



STEP 1: CLARIFY BUSINESS REQUIREMENTS

Note: This example is for the sole purpose of illustrating a technical implementation and should not be considered a source of actual medical advice or opinion.

Pneumonia is a serious life-threatening condition. Its potentially devastating effects have become even clearer over the course of the Covid-19 pandemic. It has a number of causes including bacteria, viruses and fungi. The treatment for pneumonia varies depending on the cause. Consequently, it's really important to accurately identify if the patient has pneumonia and if so, what type, so that the right course of drugs can be administered. In a busy, pressurised hospital setting, unexpected events can lead to resource constraints which in turn can lead to delays in diagnosing pneumonia. The use of a machine learning model to help with a timely diagnosis could help to save lives and reduce the burden on healthcare practitioners. A machine learning model that is capable of making high confidence predictions (that is embedded in a system with appropriate checks) has the potential to reduce costs and delays associated with blood tests and extensive consultations with specialists. Obviously, lower confidence predictions will still follow the conventional diagnostic workflow. The business requirement in this scenario is the smooth and timely prediction of likely pneumonia cases (with minimal false negatives) while reducing the number of non-pneumonia cases that are directed for blood tests and examination by consultants.

STEP 2: ASSESS AVAILABLE DATA

The dataset for the model training comprises digitized x-ray images from a range of patients – some with various types of pneumonia and others with uninfected lungs.[2] Digitized x-ray images are fairly widely used and the data format is well understood. This dataset needs to be made accessible to the data science team for model training. The sample project uploads the training image dataset to the connected data store (for CML this could be an S3 bucket). This could also be implemented using the Cloudera Operational Database (COD), as its Apache HBase Medium Object Storage (MOB) feature means it is well-suited to serving images.

STEP 3: DEVELOP A DATA SCIENCE PLAN

The next step is for the data science team to explore the dataset and come up with the plan for model development and mode of deployment. The business requirement is to reduce the time to get a diagnosis, and minimise the use of specialists and/or additional blood tests. This needs to be done with the view that the model should also make the fewest possible number of false predictions that a patient does not have pneumonia when they actually do. The model supports this requirement by minimizing false negatives as much as possible and optimizing the accuracy of the classification. A reasonable plan for achieving this would be to create an image classifier using transfer learning on one of the new generation, pretrained image classifier models. So our machine learning solution will comprise two specific models: one model capable of predicting if the patient has pneumonia and a second for predicting the type of pneumonia. The first model needs to be optimized to reduce the number of false negatives (high sensitivity / recall). Adjusting the threshold for classification into either group should minimise false negatives. The second model needs to be optimized for accuracy as the requirement is for more certainty as to the type of pneumonia. This is likely a complex computation task that will require many nodes of GPU during the initial model training process and would be best implemented using a public cloud-based CML with GPU nodes for the duration of the training and optimization processes.

STEP 4: MODEL DEPLOYMENT

A production version of this model would involve a pipeline that captures new images from an x-ray that is flagged by the radiologist as requiring pneumonia identification. This image (or in some cases multiple) of the patient's lungs would be sent to an API endpoint in CML to provide a prediction from the classifier. The data from each call to the API needs to be stored to calculate overall model performance and any result that is below an acceptable confidence threshold needs to trigger an alert to the attending medical practitioner to then either request blood test or request human assistance from someone with expertise in this field. Of critical importance here are availability and history. The model must always be available and the lineage of the data on which it was trained must be tracked in order to support auditability and reproducibility requirements. It is also necessary to store all predictions made by the specific model deployed and the detail for the image used. This allows the ability to confirm that the specific deployment will make the same prediction given the same image and to cover the Auditability requirement listed in section 4. CML provides the ability to track the image details, the prediction and a unique identifier for the model used. CML also keeps copies of previously deployed models so they can be redeployed for testing if required.

STEP 5: MODEL OPERATIONS

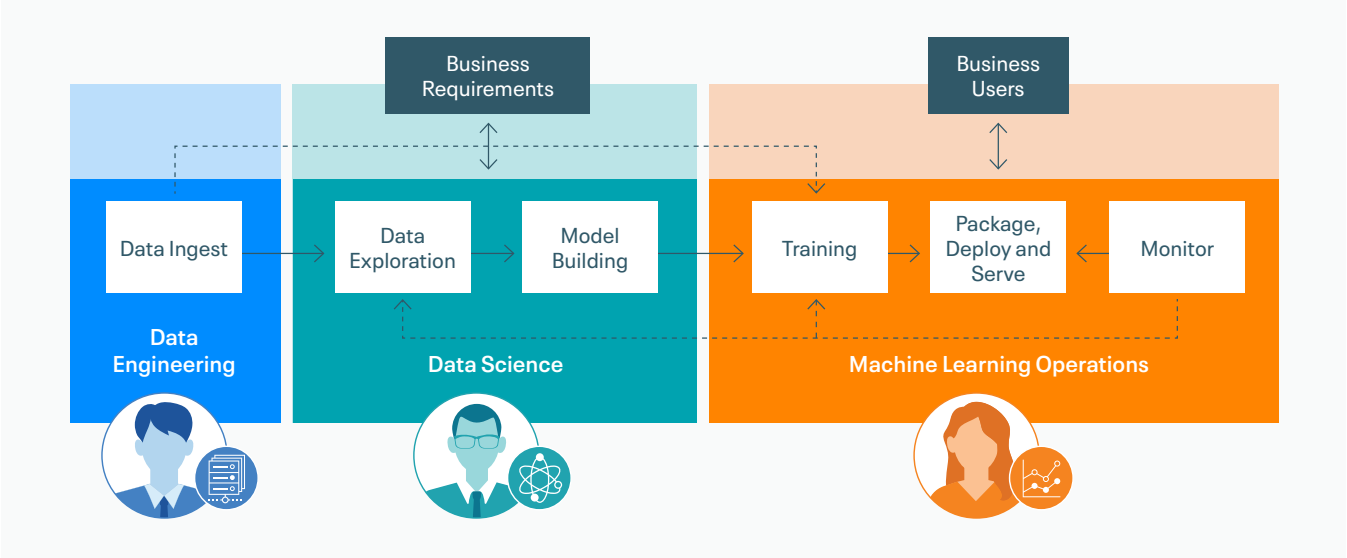
Once the machine learning models are deployed, the classifier's performance should be validated against human assessments and blood tests. Further, a random sample of the high confidence predictions made by the model should also be validated in this way.

In a scenario such as this, where the number of positive cases of pneumonia are low relative to the total number of people x-rayed, an imbalanced classification could be a complicating factor. It could result in a situation where a model appears to be making accurate predictions (and by extension, fulfilling business requirements) but in reality it's simply reflecting statistical probabilities.

In order to better understand this point, let's assume 1 in every 100 patients has pneumonia. This implies that 99% of x-ray images will have no indication of pneumonia. If our model simply classified every x-ray as normal (i.e. it never detects pneumonia) then most of the time the model would accurately classify x-ray images, it would be wrong in only 1% of cases. This would superficially satisfy the business success metrics as the number of cases flagged for corroboration through blood tests or manual assessment, would fall. This would in turn reduce the time and costs associated with pneumonia detection. However the real life implications of failing to identify a case of pneumonia would be very serious. Once the problem was eventually detected the project would, rightly, be considered a failure.

The scenario described above shows why it's so important for the design and maintenance of machine learning systems in production, to go beyond simple and efficient automation. It highlights the requirement for both statistical or mathematical capabilities and deep domain understanding and experience to deliver the benefit that a well implemented production machine learning process can bring.

Cross Functional Teams



The reality is that this kind of workflow requires input from different teams within the business. In other words, it introduces a cross-functional team requirement:

- The **data engineering** team needs to make sure that the data is clean, up to date and available.
- The **data science** team then needs to perform data exploration and model building.
- The **machine learning operations** team needs to manage the model post-deployment and make sure that it is always available, operating accurately and continually accessible to the relevant business users.

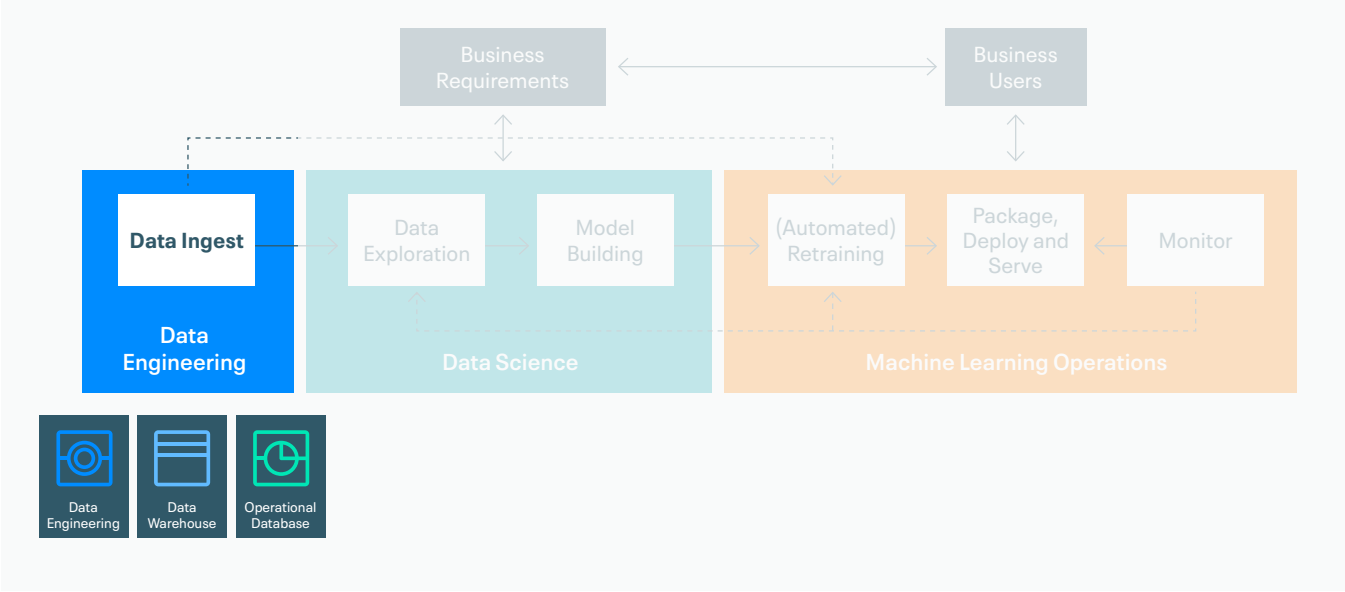
It is reasonable to expect that different parties within the cross-functional teams might have differing and strongly held opinions about how things should be done. Unless there is careful coordination, it is possible to inadvertently create the conditions for 'The Great Circle of Blame'.

This circle of blame describes a set of behaviours and attitudes that can arise within cross-functional teams wherein blame for things going wrong is attributed to colleagues based in other teams. For example, data scientists might blame their data engineer colleagues for not providing the data in their preferred format or that is not available. Data engineers might blame the data scientists for not adhering to corporate security standards during model development. ML Ops practitioners might express frustration about data scientists not writing tests for their models and data scientists might be exasperated by ML Ops refusal to accept Notebooks as working models. The Great Circle of Blame might look different in different organisations but if left unaddressed it can be a contributory factor for machine learning models failing to make it to production. In addressing this issue, it is important to develop agreed ways of working and to select a platform (and tooling) that is capable of supporting all members of the cross-functional teams to work in the agreed ways.

3: Understanding the ML-specific Infrastructure

This section highlights which aspects of your infrastructure, platform and tooling are required in order to support the cross-functional teams involved in delivering the machine learning workflow described above. The details provided here are specific to the Cloudera stack but the principles are not. These ideas can be applied to other platforms or a bespoke build comprising various components from the Open Source Software (OSS) world. It's worth noting that this latter case presents its own set of issues.

Data Access



The starting point for all machine learning projects, after the business requirement, is data. The first thing to consider is what data is already being stored by the organisation and whether it is applicable to the project. If the available data fits the requirements, then the next consideration is how accessible it is to the people and processes that need it. However, if there are gaps in the data, then it might be necessary to acquire additional data. This can come from internal data sources which aren't collected or stored in an accessible manner. This data could also come from an external third party provider. However the data is acquired, it will all need to be accessible to the people and processes that need it within the overall machine learning workflow.

Another point of consideration for the organisation is whether there is value in storing the products of intermediate data processing, such as features (representations of data that can be ingested by models), within a feature store.

Feature Store

At present, there is no consensus on the term feature store. In its simplest form it can be considered a store for curated features so they can be reused. In its more complex manifestation, a feature store might go beyond facilitating access to features and also seek to deliver high availability and low latency in production as well as more complex requirements.

In machine learning, model training requires the inputs for the model training process to be represented as numeric values. As an example, categorical variables like red, green, and blue must be transformed via a process like one hot encoding. Relatedly, certain machine learning algorithms work better when numeric inputs are scaled or normalised. Scaling is the process of taking a column of numbers that ranges from, for example, 1 to 1000 and creating a new column that ranges from 0 to 1 (often referred to as normalising) or -1 to 1 (often referred to as standardising). This represents the final stage of the data preparation before it can be used in the machine learning model training process. This feature data is not necessarily useful to other users of the data platform outside of machine learning model training though.

In some machine learning implementations feature creation is incorporated into the model training process using a 'pipeline' i.e. an automated sequence of steps. This pipeline will create the features needed for the model training process. These features are ephemeral in nature – they are discarded once the model training process is done. However, some workflows allow these intermediate features to be more permanent by storing them in a feature store. There can be benefits to doing this. For example it can help save on the compute resources required or speed up the process of hyperparameter optimization by removing the need to recalculate the extracted features for each model training run. Feature stores can also help with reproducibility. We'll cover this in a bit more detail later in this white paper.

From a data engineering perspective, Data Access is primarily focused on making data accessible by the data science and ML Ops teams. The data needs to be clean and as up to date as is necessary to meet the business requirement. Within the Cloudera stack this would involve making the data available via CDSW/CML. This could be in the form of files in the HDFS or the cloud object store, a Hive database in a DataHub, a virtual Data Warehouse in a Cloudera Data Warehouse (CDW) experience or even in HBase within the Cloudera Operational Database (COD) experience. In a DataHub the options include Druid or Kudu as the underlying storage.

The decision about what to use depends on the particular machine learning project and the data used for it. It can be tabular data, or document-like key-value data, or binary data like images and audio files. Keeping data within the stack minimises the complexity of security and governance of moving data between disparate systems.

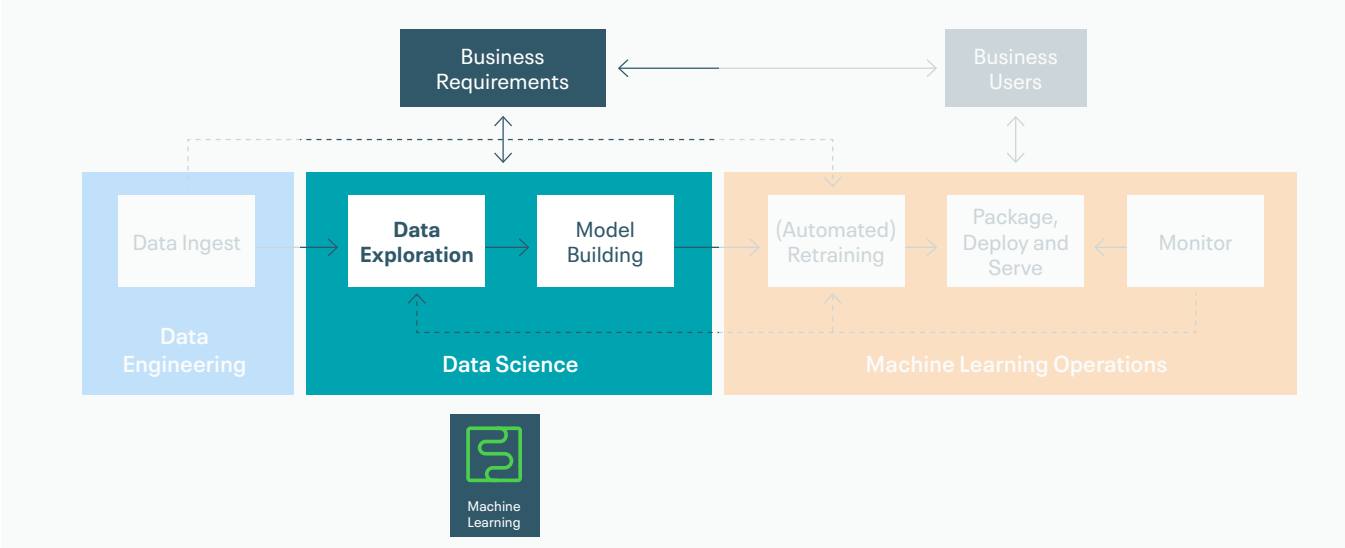
In CDP the follow storage systems are available and are best suited to certain data structures:

DATA STORAGE SYSTEM	DATA STRUCTURES
COD (HBase)	key-value data (i.e. document storage)
CDW (Hive / Impala / Hive LLAP)	tabular data
Data Hub (HDFS / Cloud Object Store)	unstructured data (images / audio)

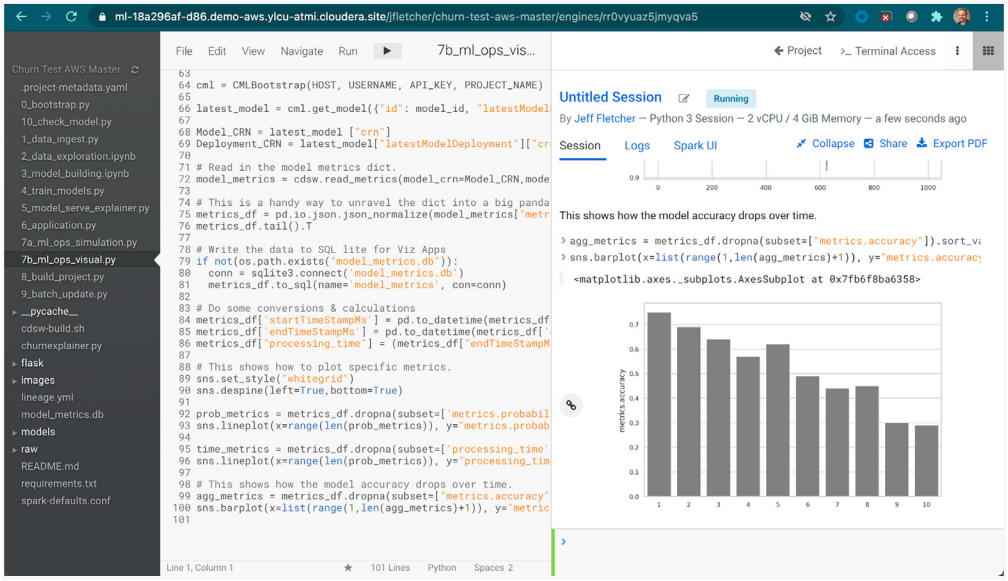
Putting extracted features into a feature store raises the question of responsibility. There are no hard and fast rules about this. Our advice is that the relevant teams - data engineering, data science or model operations- decide how responsibility is allocated based on relevant skills and knowledge. Within the Cloudera stack, the requirements for the feature store can also be met using either COD (with Apache Phoenix) or CDW for tabular data or COD for unstructured, key-value data. Having said that, the one place that makes the most sense is HBase as it supports a wide range of dataset types with low latency lookup and also has better snapshot capabilities for reproducibility.

Regardless of which CDP experience is used to make the data available to the data science team, CDE is configured and optimized to do this at scale and in an automated way that should eliminate friction between all involved parties.

Data Exploration



Exploratory data analysis (EDA) is a fairly well understood function within the data science community. CDSW/CML supports EDA via its native workbench as well as third party notebook solutions such as Jupyter Lab and R Studio. It is at this stage that the data science team will look at the 'shape' of the data and come up with a plan for building a suitable model. EDA is often combined with the Model Building stage, however this doesn't mean a single team necessarily manages the entire process. It may be that there are good organizational reasons to split this activity, for example to accommodate areas of specialization. Some personnel may be more inclined towards statistical data analysis and so focus on EDA while others might specialise in model optimization and concentrate on that part of the model development process.



What is most important is that at this stage the data science team should have easy access to all the relevant data and have the flexibility to augment that data (if necessary) in order to best serve the business requirement.

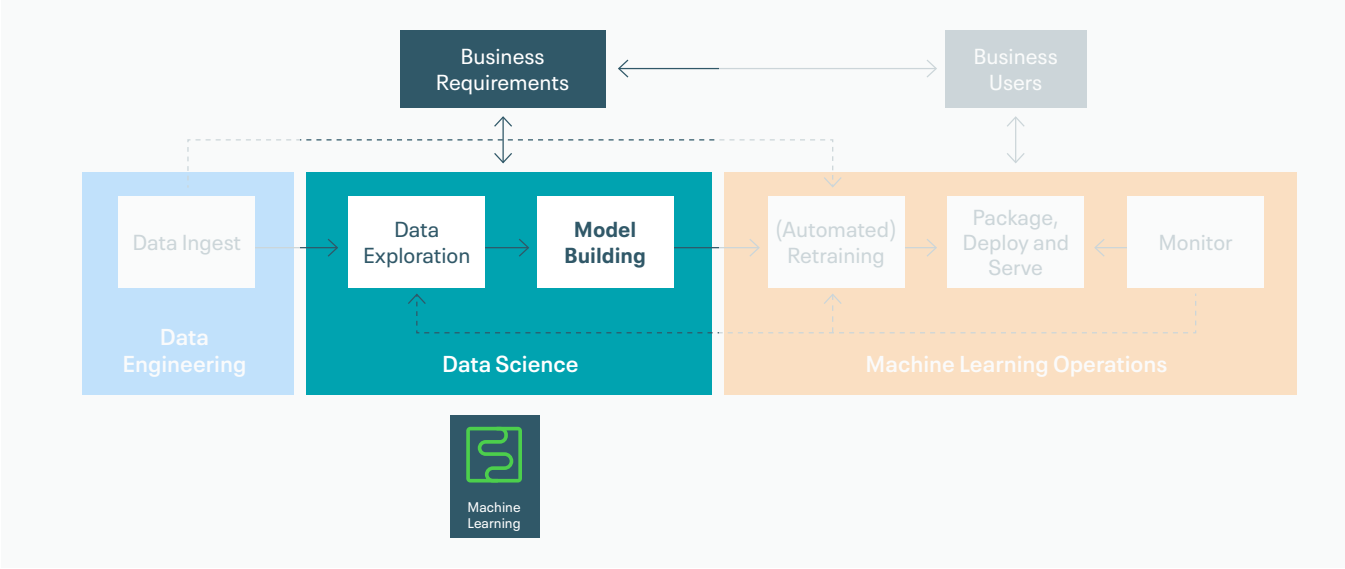
Source Control

One of the main parts of the artifacts created during the various stages of deploying a machine learning project is the actual code written by the data scientists and machine learning engineers. Creating code usually starts once Data Exploration starts, but there may be some code that is used to facilitate or create data that is needed for data access. These code artifacts need to be persisted and stored and have the ability to keep track of a history of changes. This is required for the auditability and regulatory requirements discussed later, but also for the developers to be able to collaborate effectively and roll back if any changes are made that cause issues with the project. This is where source control comes into play. Like any other form of software development, not all changes to code are significant, but anything that is considered significant or just as a way to store the current state of the code and other software artifacts, source control tools are invaluable.

Source control systems are a solved problem, with many feature rich, tried and tested tools available. CML/CDSW seamlessly integrates with any git compliant source control system: Github, Gitlab, Bitbucket etc. Some source control systems have workflow automation tools built in which can integrate with the CML/CDSW APIs, such as the Jobs API to further enhance workflow automation.

The main thing to keep in mind is that source control is used to store the code. It can be used to store other artifacts, like saved model files or data sets, but it's not designed for that. Source control systems should be used as part of the overall platform to manage version control for software code.

Model Building



While model building is often done in conjunction with the previous step, bigger teams might support some degree of specialisations of roles. Some team members will focus on implementing the best approach to addressing the business use case while others might concentrate on model selection and optimization. For the sake of clarity, let's confirm what is meant by **model** here. The code used to create the model will likely include a current set of optimizations and hyper parameters specific to the model. This code and any iterations used to create a new version of the model that contains changes to the way the model be trained should be checked into the source control system.

What is a model?

The term model has many uses within different contexts and its meaning can change even within the data science context. For the purpose of this document a model is:

> *"A combination of a data transformer, an algorithm and configuration details that can be used to make a new prediction based on new data."*

Regardless of whether this is implemented as a single step, a couple of steps that involve getting the transformed data to and from a feature store, or an activity within a pipeline, the definition still holds. The final form of the model is an artifact.

Model Formats

The final trained model artifact is a file that can be loaded at a later date and be used to make new predictions. This file can exist in a variety of formats including:

- **Pickle**: a standard python serialisation library used to save models from scikit-learn
- **Spark MLWritable**: the standard model storage format included with Spark, but limited to use only within Spark
- **PMML**: (Predictive Model Markup Language) a standardised language used to represent predictive analytic models in a portable text format
- **ONNX**: (Open Neural Network Exchange) provides a portable model format for deep learning, using Google Protocol Buffers for the schema definition

Model optimization

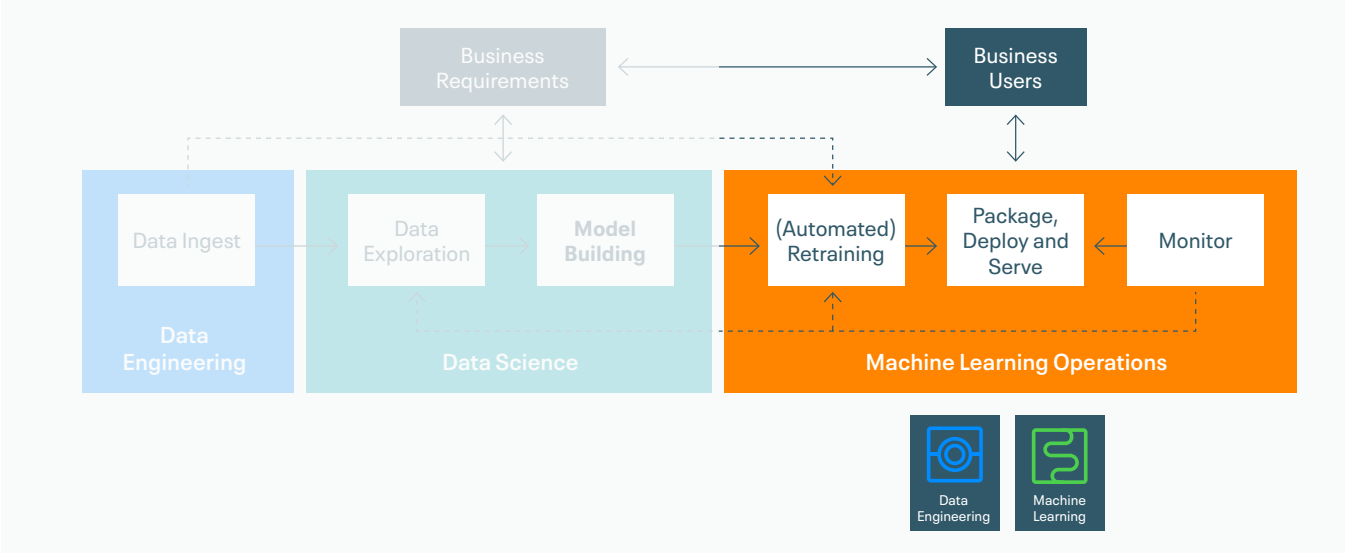
The initial model building process incorporates some work towards improving model performance through hyperparameter optimization and some exploration of different approaches to feature creation. The Experiments feature within CML/CDSW supports this model optimization work. Alternatively, this can be done using some of the available model optimization OSS tools such as TPOT, auto-sklearn, MLFlow etc. Apache Spark does not have any built-in tools for model optimization, so the CML/CDSW Experiments feature would be a good option in this case.

AutoML and other automated hyperparameter optimization tools are useful but should be considered in light of available compute resources and machine learning skills. A data scientist with enough experience and domain expertise will probably know what to do to get to an optimized model more quickly and cheaply than an AutoML tool would. For many use cases an AutoML tool is no replacement for an experienced data scientist, who will be able to develop a more effective and interpretable model. Consequently, it should not necessarily be included as part of the process for all new models if it doesn't need to be.

The infrastructure needed for doing model training is another point of consideration. The next section explores this further but it's worth noting even during the early stages of model building, model optimization will require several iterations of model training. The type of model, the size of the data and available compute resources will dictate what kind of infrastructure is most suitable for running the model training process. For example, training a big neural network on a very large dataset will require multiple GPU nodes to complete in a reasonable amount of time, but this is expensive infrastructure. In this situation, training needs would be better fulfilled using CML in the public cloud to take advantage of its ability to autoscale the number of nodes required to meet the compute demand. If this were a smaller model training process or public cloud was not available, then using an existing on-premise CDP Base cluster with Spark on Yarn would make more sense.

Once the initial model build is complete and there is a working process for training the model, this needs to be automated so that the model can be periodically re-trained without direct input from the data science team.

Model Training



As part of the ongoing production machine learning workflow, the machine learning models will need to be periodically retrained. This may be because new data becomes available, the data landscape changes or a newer version of the algorithm is released that optimizes some part of the models' performance. Having the right infrastructure becomes more important here as this often needs to happen without direct involvement from the data science teams. The machine learning operations team needs a way to run a model training process directly on the available data. This may include a pipeline that does feature creation as part of the model training, or separate processes that put the features into a feature store first. Where the model training jobs sit will depend on the use case.

These model training jobs need to be automated in a way that they can be triggered through the monitoring system (discussed later in the document). The requirement is that model training jobs are schedulable and triggerable through an API call from a process running as part of the ongoing model monitoring system. This can be done entirely within CML/CDSW using the Jobs feature, or if this is a larger Spark based job it can be done in CDE. The public cloud implementation of CML uses Kubernetes to scale up and down the resources that are available for model training. This allows it to support model training jobs that need a lot of capacity for a short period of time.

A hybrid cloud approach would probably provide the optimal price to performance ratio for organisations that need to manage model training for a variety of models with a range of infrastructure requirements. Having said that, there are still complexities around data locality and its impact on efficient data access that need to be understood and managed. The economics of hybrid cloud machine learning is discussed in further detail here.

Package, Deploy and Serve

The trained model now needs to go into **production** i.e. the model's output needs to be 'served' to or integrated into downstream applications. Depending on the availability requirements discussed later in the document, some organisations run completely separate environments to differentiate between development and production systems. Once a model has been developed and tested on the development system, it is moved on to the production system for deployment. This allows for different architectures and configurations between the two platform types to meet differing compute and availability requirements. Within CML/CDSW, this development-to-production capability can be facilitated in a single workspace by using the Teams feature. Alternatively CML/CDSW can be deployed with completely separate workspaces in the same or different environments (for CML) or different hardware (for CDSW) and the projects can be moved using the underlying source control system and redeployed. For more automation, the projects can use the CML v2 API or AMP specification which will allow for automating model and job deployments as well.

The two main modes of operation for production machine learning models are **batch** and **near realtime**. There are other modes of operation for machine learning models, but they are far less common.

Batch Models

A batch process runs periodically and will make several inferences at a time. It might be a process that runs batch inference on tabular data and adds or updates a column with predictions made by the model. These kinds of models are supported in CML/CDSW using the Jobs feature to periodically trigger a script that can fetch new data from and update one of the datastores within CDP (e.g. COD or CDW) with a new prediction. As illustrated in the churn projects, the model metrics are updated during the batch process, but if model metric tracking is not required, it's possible to just update the table on each new batch run.

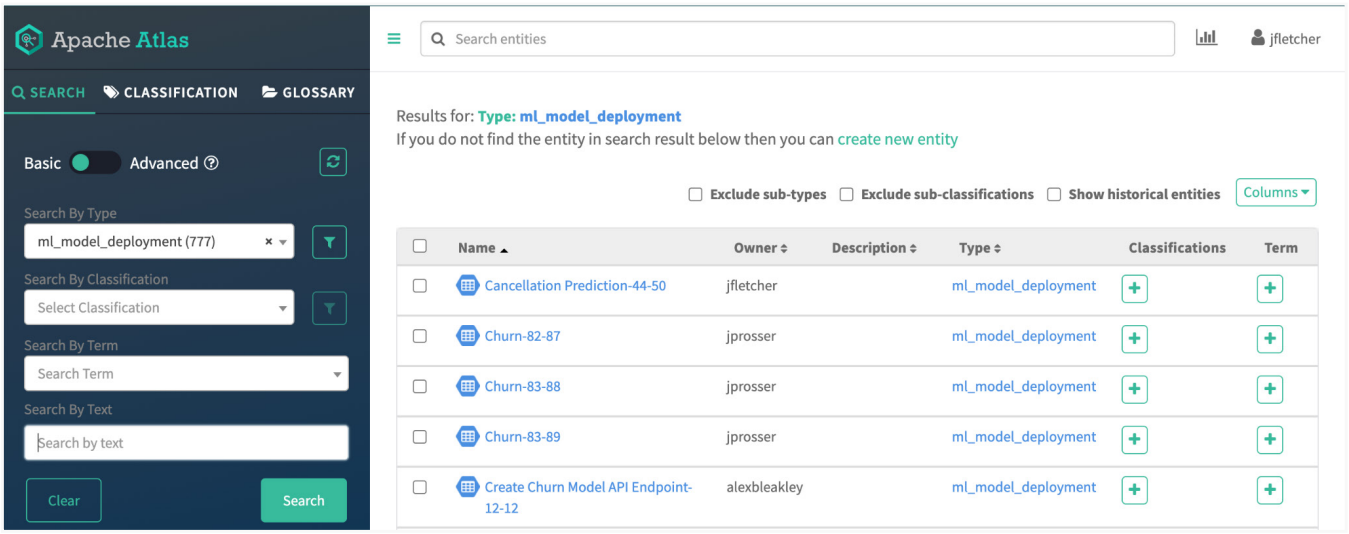
Realtime Models

Realtime models' output are usually made available via APIs that can be called by downstream relevant applications in order to make new predictions on an ad hoc basis. This is the default mode for the Models feature within CML/CDSW. The model artifact (i.e. the model file) is loaded as a Python function which is then wrapped in a process that creates an API endpoint which can accept JSON as input data, run the model code to create a prediction and then return the prediction as JSON via that same API endpoint. This process is usually referred to as model serving.

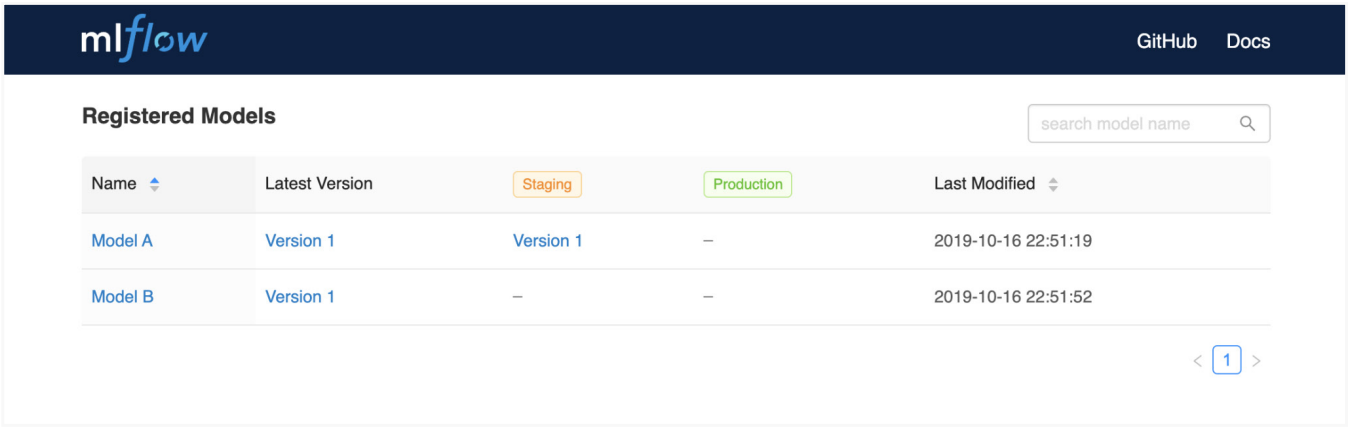
Model deployment and serving infrastructure have specific availability and reproducibility requirements. These are detailed in the section on continuous operations for production machine learning.

Model Registry

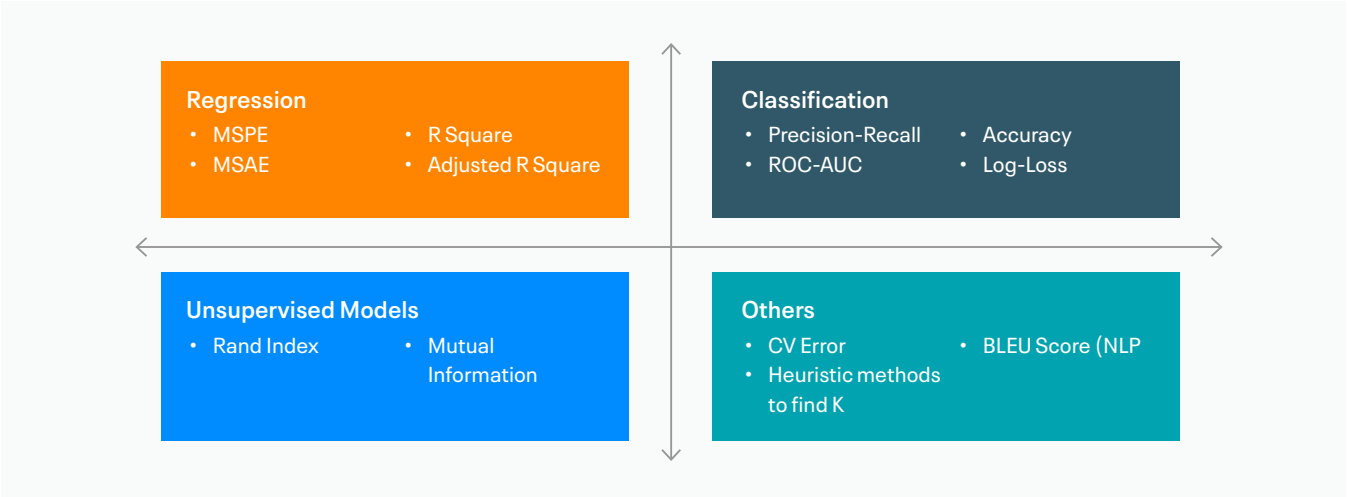
A Model Registry is a mechanism that tracks and manages models, model artifacts and related to metadata for any deployed model type. In CML/CDSW, there are 2 options for a model registry. There is the model governance feature, which registers the various stages of model deployment with CDP's Atlas:



Alternatively it's possible to use MLFlow within a CML/CDSW project, but this will be local to the project and not registered globally like it would be with Atlas.



Monitoring Systems



The final stage in this machine learning workflow supports the monitoring of the model. This aim is to ensure that the model output is of a consistent performance level and reliably available to the downstream applications that need to consume it, as per the business requirement. The monitoring-related tasks are automated and delivered with the aid of optimized tooling and infrastructure. For example, within CML/CDSW it's possible to check and monitor the real-time model to ensure it's up and working.

For the metric tracking requirements there is an SDK that allows the model operations team to record any number of relevant metrics each time the model is called. These metrics can be updated to include real-world data for supervised learning models and the periodic addition of aggregate metrics to track any required statistical performance metrics over time.

As will be discussed later, it's important to know how the model is performing against both the statistical and business requirements. When monitoring a model, many implementations will focus on the statistical metrics and how these vary over time. However, this is insufficient. Many projects fail because the business requirements they were designed to deliver aren't being met . Take for example the churn example accompanying this report, the fact that the model is 95% accurate at predicting churn is not useful if this doesn't translate to measurable reduction in actual churn rate. It's achieving the latter –the business requirement – that actually delivers value to the business.

All of the above requirements can be implemented using components from CML/CDSW and other experiences within the CDP stack. It is possible to build an implementation using open source components provided the data security and governance requirements can be fulfilled too. While these two criteria don't have a direct impact on model performance, they are important concerns for an IT department; it will also result in fewer integration issues down the line. An end-to-end ML solution that lacks robust data security and governance is unlikely to be accepted by an enterprise IT department.

4: Understanding Continuous Operations for Production Machine Learning

In this section we will outline some specific implementation methods and actions to take when architecting and implementing production machine learning systems. We call this methodology Continuous Operations for Production Machine Learning (COPML). This is not just about automation of model tracking, it also includes the various factors that need to be considered so that machine learning projects continually deliver successful business outcomes. This is a methodology by which enterprise users can maintain production machine learning systems in place with the minimum amount of input human input and while also adhering to the specifics and the more intangible requirements for this production machine learning project.

Why not just use Continuous Delivery?

"Continuous Delivery is the ability to get changes of all types — including new features, configuration changes, bug fixes, and experiments — into production, or into the hands of users, safely and quickly in a sustainable way".
— Jez Humble and Dave Farley

It's worth asking why an existing framework like continuous delivery (CD) can't simply be adapted to work for production machine learning. Software engineers and teams that use CD effectively, will know that it provides an easy and efficient way to iterate and update software. In fact there are other approaches to conscientious operations for machine learning that take this approach: See here.

However, there are some fundamental requirements that can't be addressed through this process. CD may be able to keep models operating, however, it lacks the flexibility to address the less structured and clear-cut aspects of sustaining a project. The COPML methodology on the other hand is far more accommodating of necessary interventions to the system, whether these are automated or manual. Understanding when and how these interventions need to happen is an iterative process. In other words, there needs to be a period of initial adjustment during which optimizations can be made. CD cannot accommodate this.

Another issue that makes CD a less than ideal approach for managing and sustaining machine learning projects in production is that the CD methodology has been optimized for software systems. However, data science teams are not primarily software developers –their interests and concerns are broader than the proper working of the code. They, necessarily, have to take a whole system view. Consequently, they need a methodology that allows for continuous operations and continuous optimization by people who are as skilled at understanding the output of the system as well as the statistical metrics underlying them.

COPML Overview

This methodology has two main success criteria, the fulfillment of business requirements and regulatory requirements. It costs money to get the infrastructure, software and people needed to implement effective machine learning applications and systems. Therefore, one important metric for success is a quantifiable return on that investment. Business requirements focus on indicators related to this outcome.

The regulatory requirements relate to implementation considerations which ensure that the machine learning project is compliant with all the relevant legal, industry and ethical standards. In the next section we consider these two requirements in further detail and discuss how the COPML methodology delivers against them.

Business Requirements

As already discussed, a higher level business requirement is always the impetus for a machine learning project. There is usually an explicit metric associated with the business requirement, for example, in the churn reduction scenario described above, this metric was the churn rate and the target was for it to be reduced from 10% to 5%. Consequently, the focus for the machine learning project is the correct identification of customers who are likely to churn **and the application of retention strategies**. If this fall in the churn rate is achieved it is tempting to rule the project a success. However, it's possible for a project to satisfy its primary business metric but do so at too high a cost. If this happens the project will still be deemed a failure. This is why it is important not to over-capitalise these projects. Taking the following steps will help keep running costs down.

Availability

Availability in IT terms is a reference to infrastructure uptime. If the machine learning model is able to make new inferences, it's considered available. High availability design for IT systems is a well understood and documented field. The requirement here is to align the availability for the deployed machine learning models with the needs of the relevant business processes. Assuming the business is getting value from this machine learning model it needs to remain accessible, but there is context that needs to be set according to value derived from the machine learning model.

In the churn example, the batch implementation does not require compute resources to be deployed continuously for the model to be useful. A simple implementation would probably run once a week, a table is updated with the relevant predictions and the retention process can be managed manually, using this data. Therefore the availability requirement can be satisfied by the underlying CML/CDSW platform that the job runs on. Using CML in the public cloud with the Jobs function will keep the costs relatively low while still meeting the business objective.

For the pneumonia example however, the real-time model needs to be highly available, every day of the week. For a real-life use case with this level of importance, an uptime or availability metric might be set as high as 99.9% or 99.999%. In this instance real-time CML models can be deployed using replicas and availability metrics set in the public cloud to ensure near continuous availability of the API endpoint. However, in the case of especially high availability requirements, it would be best to implement a backup CML workspace in a different cloud provider or region, running the same model, also using replicas and have any applications fail over to using the backup model API.

When considering the availability requirements for any production machine learning project, the key decision is the balance between cost of the infrastructure required to make the model available and the business value that the model provides.

Action Items:

- Decide on the required availability for the model based on the business objective
- Create an architecture to support this availability requirement

Effectiveness

Once the machine learning model is in place and meets the availability expectations, it's necessary to create measurable model performance objectives to evaluate the effectiveness of the machine learning model. The reality is that most machine learning projects implemented in an enterprise context are fairly mundane and lack the fanfare of beating a world chess or Go champion. However these projects still need to be performing at a high enough level for the business to derive benefits from the money invested into it.

Creating a measure of effectiveness requires well-defined success metrics for the outcomes of the machine learning model implementation. These metrics should cover both the statistical/mathematical performance of the model and its satisfaction of business objectives. We have seen a tendency among data science teams to focus very heavily on the statistical measures of model performance. While this is important, it's not necessarily the most important yardstick of success.

For the churn prediction example, the business set a churn rate target i.e. to drop the monthly customer churn numbers from 10% to 5%. The model can be measured statistically using one of the binary classification evaluation methods available. If the chosen measure is accuracy, the model's statistical performance will be its ability to accurately predict which customers will churn and which won't. The example project creates a model that has an accuracy measure of 80%. Assume that once the model is in place, those customers who are predicted as likely to churn are put through some retention process that keeps the customer rather than having them churn. This will have two effects: Firstly this will reduce the churn rate for that month of measurement, let's say from 10% to 7%, but it will also reduce the accuracy of the model. Those customers who were predicted to churn have gone through the retention process and will now stay. Therefore when measuring the model performance at the end of the month, the accuracy could drop to 75%. But the model is still effective. However it would now be prudent to retrain the model as some customers will have churned despite the retention process and the model can now be optimized to better identify these customers and possibly implement a different retention process.

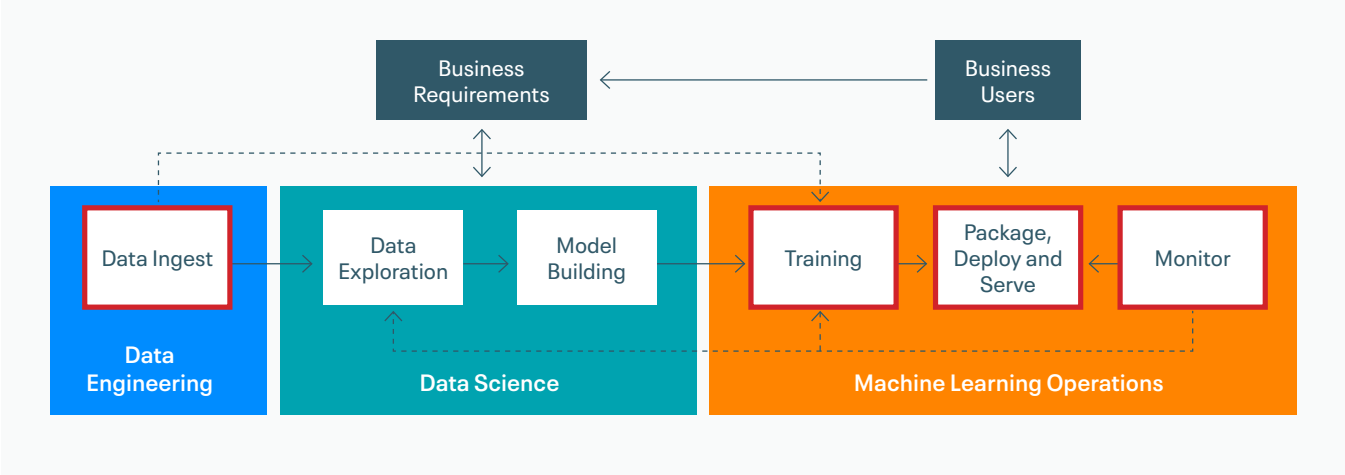
For the pneumonia model, the key metric is the false positive rate as the consequences of an inaccurate diagnosis are severe. This means optimizing the model for sensitivity. In the example project the first model, which is used to detect if the patient has pneumonia, currently has an accuracy of 97%. The second model, which is then used to predict the type of pneumonia needs to be optimized for accuracy. The example project has an accuracy of 80%, which will be considered the minimum accuracy level for the project. Both measures need to be tracked over time, but this is done in batches where the outputs from the models are then compared with the real-world values to check the model performance. The key business metrics to track are the number of false negatives the first model produces and the number of times either model requires human intervention to confirm a prediction. Tracking these values over time will determine the overall effectiveness of the whole project.

This process of continuous testing and retraining is outlined in the next stage, but for now the requirement is to create an effectiveness measure that allows for an evaluation of the overall machine learning project with the business objectives.

Action Items:

- Set up agreed measurements for the effectiveness of this production machine learning process. These should include both statistical measures for explicit model performance and business metrics for overall performance. This can include costs of infrastructure and people involved as even a well performing machine learning model may simply cost too much to be of long term business benefit.
- Create trigger points for these effectiveness measurements and actions to be taken when a threshold is reached.

Automation



One of the biggest operational costs of any machine learning project will be the teams that support it. There is the direct cost associated with the various data teams working on their respective parts of the project to get things up and running, but there is also the opportunity cost of having these teams do something that could be automated rather than working on other projects. As much of the process as possible needs to be automated. Within the standard workflow described, the following components can be automated with components from CDP.

DATA INGEST (CDE)

This can be a continuous process or a periodic one but the actual implementation isn't determined solely on the requirements of the downstream applications that are dependent on the model. The large data platform requirements will also have to be taken into account.

For the churn example, customer data is always changing and many other systems use this customer data. Therefore data ingest will likely be automated anyway, outside of the need for training the machine learning models. However there might be an intermediate feature creation step that isn't part of the standard pipeline which will also have to be automated. A CML job can be run once a week that creates a feature store that will be used to retrain the model.

For the pneumonia x-ray example there needs to be an automated data collection process that puts the images and the results from the predictions made by the model(s) and any manually obtained data into a data store that can be used for model retraining.

Action Items:

- Decide on any new or additional data ingested automation tasks specific to this machine learning project that is covered by other data usage requirements. This could include creating a feature store specific for machine learning model training. Schedule these as automated processes to run on the required frequency.
- Assume the end state for this data ingestion is to make the data available to model retraining jobs as well as the data science teams for the purpose of (additional) exploratory data analysis and model optimisation.

MODEL MONITORING (CML MODEL SDK AND JOBS)

Model monitoring automation will likely follow with the availability requirement. Automated monitoring should cover the availability and effectiveness requirements of the project. These factors should also inform the frequency of monitoring.

In the churn example, customer churn is measured monthly, therefore an automated process that runs once a month to evaluate the model performance will suffice. Depending on the outcome of the model evaluation process, other tasks can be triggered e.g. model retraining, model redeployment etc.

For the pneumonia x-ray example, the model uptime needs to be monitored to ensure compliance with the stated availability metric. Whenever the model becomes unavailable, the monitoring process should raise an alert. The performance of the pneumonia x-ray model can be automatically evaluated after a certain number[3] of predictions have been made. This will also inform decisions about thresholds set for retraining etc.

Some threshold values will serve as indications of declining model performance or significant changes in the data landscape which can not be fixed purely through automated remedies. These values need to be identified during the model build and exploratory data science phases and (can be programmatically) set as additional thresholds in model monitoring. When these thresholds are breached, the subsequent automation tasks should include an alert or notification for human evaluation. In the churn example, the model accuracy dropping to 70% (or lower) for three consecutive months with a customer churn rate that remains at 10% (instead of falling), is considered a performance failure scenario. If the automated model retraining and redeployment interventions fail to improve the model performance, then this is a trigger for an evaluation of the entire project.

Action Items:

- Decide which metrics (such as granularity and frequency) need to be monitored as part of this project informed by insights gleaned from the availability and effectiveness steps.
- Implement the monitoring processes and create triggers and actions should any metric cross a threshold for these values.
- Consider how re-evaluation of the ML project/application will be triggered and which automated tasks should trigger this and at what threshold.

MODEL RETRAINING (CML JOBS)

In a constantly changing data environment, the performance of machine learning models can 'drift' and become less accurate (and by extension, less value-adding) over time. The Model Monitoring section of the workflow addresses requirements related to tracking and detecting changes in performance. It also supports some automated mitigations for drops in performance e.g. model retraining. This is one of the easiest parts of machine learning automation to implement, because the model training process has already been created during the original model build. Provided the data is accessible, whether in raw form, or in an intermediate form via a feature store, most machine learning platforms will support automated model retraining. The CDSW/CML Jobs capability allows for model retraining automation with variable infrastructure requirements.

For both example projects, the model retraining is done using the CML/CDSW Jobs function. This is done using 2 separate jobs. The first job will check the current metrics stored with the deployed model using the model metrics feature. If the value of the metric drops below a threshold, a second job is triggered that will perform the model retraining and redeployment.

For the pneumonia x-ray example, a real world implementation using tens of thousands of images and a more complex pretrained model, model retraining could require 100s of GPU nodes for a few days to run. CML in the public cloud can facilitate this rapid scale up and scale down capability.

Action Items:

- Decide on the infrastructure needed for model retraining and the likely frequency.
- Implement an API based automated system that can launch and run the model retraining process from a task within the model monitoring process or any other external process.

MODEL RE-DEPLOYMENT (CML MODELS AND JOBS)

Once a model has been retrained, this new model needs to be deployed. The model deployment (i.e. real-time or batch) will dictate the type of automation that will need to take place.

For the churn example, when implementing in batch mode, this automated process will be to replace the model artifact used during the batch run. The model training job in the sample project overwrites the file used for the batch prediction job.

For the pneumonia x-ray example, this is a live model and needs to adhere to an uptime requirement. The example project redeploys the newly trained models to replace the existing live model API endpoints. For a real world implementation, the new model end-points should be deployed and tested using the newly trained model first and then the overall pipeline can be switched to this new model. CML provides for this automation through the model API, but monitoring and keeping track of availability should be maintained during this process.

Action Items:

- Create an automated task that will deploy the new model into production and decide on how it will be triggered.
- Maintain model monitoring for real-time models that have an uptime requirement

Risk Management

From the perspective of the business, the success or failure of the machine learning project is often viewed in terms of the project's ability to deliver a return on investment. The assessment of risk can be quite narrow and may not be considered in regulatory, security and reputational terms. Issues of security and reputational risk need to be considered and addressed as part of the larger project implementation (regulatory risk is addressed in the next section). As is the case with any applications within a data platform that handle personally identifiable information (PII), machine learning models need to be appropriately secured. This is true even though the model does not directly ingest PII. There are real risks of inadvertently leaking sensitive data if the model is accessed by malicious actors e.g. using so-called inversion or membership inference attacks facilitated through surrogate models.[4]

Another source of risk for a machine learning project lies in unidentified biases that are inherent in the underlying data used to train the models. This type of risk can be hard to detect and quantify. This is why we recommend that for any significant business investment in machine learning, organizations should consider commissioning an external auditor with experience with these kinds of projects. There is also newly proposed EU legislation (pending review) that will limit the purposes to which machine learning can be applied.

Note: For both example projects, it is assumed that these implementation risks are managed and implemented outside of the CML/CDSW project. Both projects contain PII which the CDP platform can help to secure in a number of ways including, for example, redaction. Neither of the example projects referenced in this document has been assessed with regards to inherent bias in the underlying data.

Action Items:

- For projects with any significant business investment is required and where there are large data sets of any personally identifiable information, the data and the security processes should be externally audited to check for bias and security risks.

Regulatory Requirements

Many countries and regions have regulatory requirements which govern the type of data that can be collected, the conditions under which this can be done and the ways in which such data must be managed. For example, organisations operating within the European Union are subject to the General Data Protection Regulation (GDPR), and healthcare providers in the US have to comply with the Health Insurance Portability and Accountability Act (HIPAA). Many of these regulations are understood with regards to data privacy and security. However there are specific conditions, especially within GDPR, that have implications for the use of automated decision making. For example, Article 22 of GDPR[5] grants the data subject the right to an explanation for decisions that are solely algorithmically-driven. This has implications for the design and implementation of machine learning applications, including whether or not to incorporate (demonstrable) human oversight, and if so, where and how. Satisfying the requirement for an explanation of an algorithmically-driven decision would involve finding the correct version of the data and machine learning model used to make the decision. It would also be necessary to explain which variables within the data informed the model's decision. Lastly, it is likely that your organization will need to demonstrate the consistency of the decisions made by the ML application by being able to reproduce the decision given the same data and model. Otherwise, a complainant could credibly claim that your ML application or system leads to arbitrary and potentially unfair outcomes. Even if your ML-enabled application or system has sufficient and demonstrable human oversight, it is still a good idea to be able to answer these questions as these are the types of questions a regulator might ask. That's why the rest of this section is dedicated to exploring how Cloudera's CDSW/CML experience coupled with the COPML methodology can help you build ML implementations that are auditable, reproducible, and explainable.

Auditability

The regulatory requirement regarding auditability for machine learning is specific to the model and data used to make a prediction. The requirement for GDPR is to know which specific version of the model made the prediction, the data for that prediction and the data used to train the model. The first two requirements are specific to auditability and in CML/CDSW, these requirements can be implemented using the model governance feature to track model versioning and data lineage (in CML/CDSW this is the equivalent of a model registry) and the model metrics feature to track the input data upon which the model makes prediction. Note: In this document we use the terms prediction and inference interchangeably, although the latter is the more academic term for the use of a model for making predictions.

The final requirement relates to the data used to train the model and is covered in the section on reproducibility.

Both of the example projects featured in this paper use the model metrics and model governance features of CML/CDSW and can therefore be considered as auditable.

Action Items:

- Ensure any machine learning models that are implemented that have a material effect on a data subject are tracked. This includes identifying the specific model and model artifact used to make the prediction and any data used for the prediction.

Reproducibility

There are two requirements that need to be satisfied in order for the output of a machine learning project or application to be considered reproducible. Firstly, any prediction made by the same version of the model, with the same input data, needs to give the same prediction. Secondly, when a new version of the model that has been trained on the same data set as previous versions of the model is given the same input data, the same parameters and the same random seed, it should make the same predictions. There might be some minor variability if a model is trained on a different architecture but this should be insignificant when comparing the statistical metrics and outputs during model testing. If a new model is trained using that same input data, parameters and methodologies as an original model and yields the same prediction results, the model can be said to be reproducible. With regards to the first condition, it's worth noting that there will be some variability in the response of certain model types but these are rare and usually not found in enterprise implementations. Specifically, if there is a model that is used for decision making that affects a data subject (i.e. the life of an actual human) the same model should always make the same prediction when provided with the same input data.

These requirements highlight the main factor for reproducibility: access to the original data used to train the model and the source code created to run the model training process. However, the size of the datasets in an enterprise implementation presents some complications. That's why the CML/CDSW model governance feature uses metadata to track lineage of the data, not the actual data directly. The data can be stored in a snapshot and added to the lineage metadata. In an academic or proof of concept (PoC) context, the data sizes used for machine learning projects are often small enough that the data can be bundled together with the model artifacts. But for larger datasets (i.e. 10GB or more), something enterprise implementations often require, bundling the data with the model artifact is impractical. If the implementation also uses a feature store, this can complicate things further if the original source data is not also stored in the feature store and there is some machine learning based preprocessing e.g. PCA that is required to be part of the process to reproduce the model. Within the CDP platform, there are multiple data storage and database options available and each has its own backup and restoration policies. Depending on the location of the data storage (e.g. on premise HDFS or in the public cloud using S3 or ADLS), the data replication and 'snapshot' management process will be different.

Given how data grows in size over time there are some decisions to be made at the start of the project regarding data reproducibility. These are covered in the action items later in this section.

The final part of the reproducibility requirement is for the parameters and configuration details used to train the model. Each training run that creates a model which is then put into production will have a set of parameters, hyperparameters, config details etc used during the training process. Recreating these models requires both the data and these parameter details. This is often implemented using version control systems like git, but these updates have to be triggered manually. Using the Jobs feature for model training in CML/CDSW and adding parameters details to the output of the job will keep a history of the model training process thus allowing for replication by others. This can be further augmented with metadata bundled with the deployed model using the CML/CDSW model governance capability that includes the details of the job that was run to train the model.

Given the high-impact of the decisions that the pneumonia detection model will inform, it is critical that it be reproducible. As this project uses a fixed amount of images and doesn't include any additional ones for retraining purposes, it does not demonstrate automated retraining. However, the process to create a snapshot in the Cloudera Operation Database experience (COD) is documented here. In a real world implementation, each time a model is retrained, this snapshot process needs to be run and the snapshot detail and model version information tracked.

While the potential impact of the churn model is relatively low, there is still value in making it reproducible. Incorrectly, predicting a customer as being at risk of churning and offering them free call minutes or data as an intervention, is unlikely to run afoul of regulatory protections. As such it's unlikely that a regulator will demand an audit. Despite this there are other good reasons for facilitating reproducibility e.g. debugging. This is especially true if the output of one model might feed into another system downstream. This type of entanglement makes a systematic approach to debugging even more important.

Action Items:

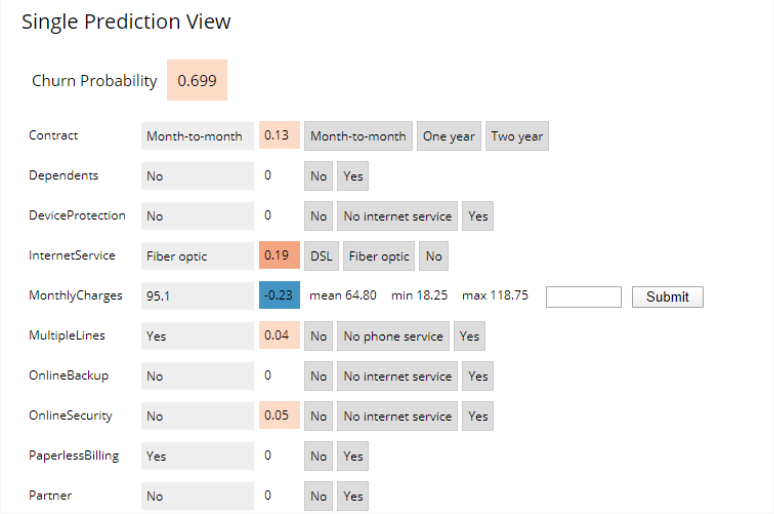
- Understand how the model is going to be used and check the extent to which the model's predictions could affect people's lives. Also, understanding the former will support your ability to assess the latter. This information will also help determine the best way to make the model reproducible. It's also worth noting that there might be regulatory implications that need to be taken into account as well.
- If necessary, create an automated process that replicates or creates a snapshot of the data used each time the model is trained/retrained.
- Create a process that will keep the details of the parameters and any other pertinent configuration information each time the model is trained/retrained.

Explainability

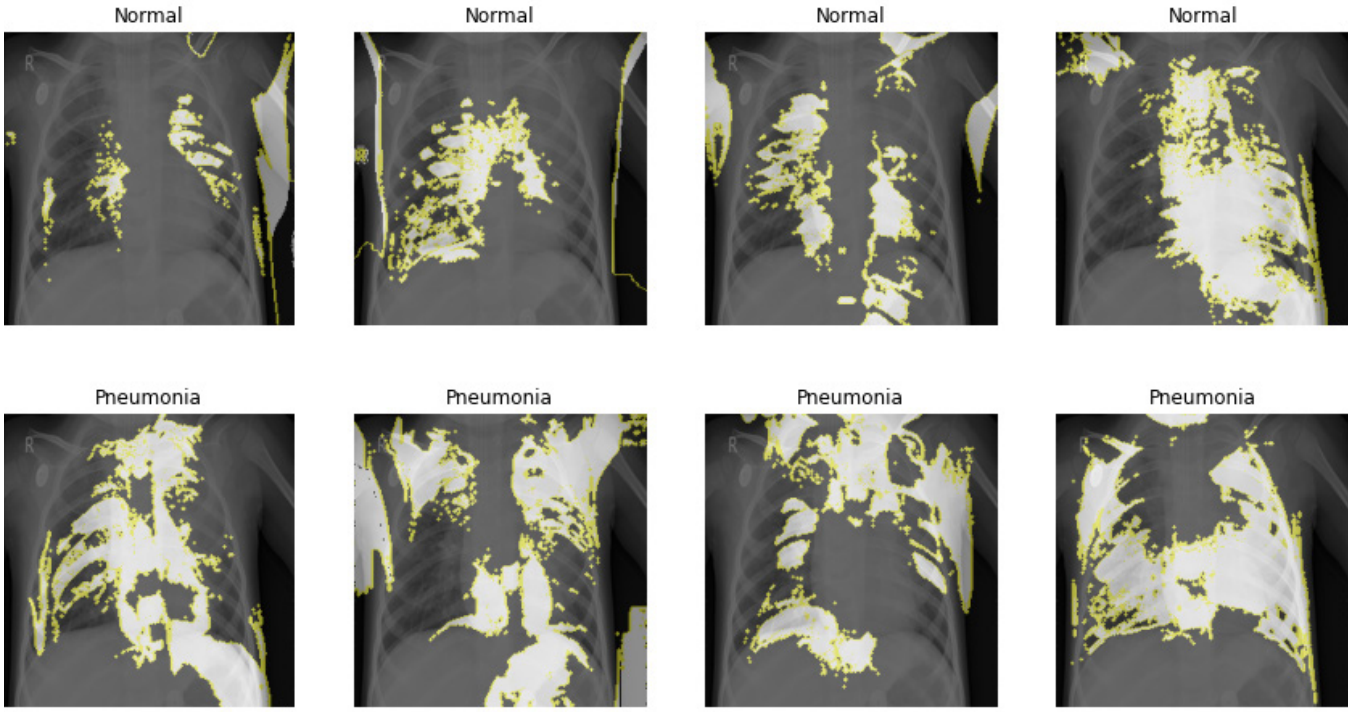
The final regulatory requirement has to do with explainability. In the context of GDPR, the data subject has the right to an explanation for a decision, if it was solely algorithmically-driven. This is perhaps less well defined and understood than having a system that is auditable and reproducible as the notion of explainability is less concrete and measurable. But the consideration here is for facilitating the understanding of the person impacted by the model-driven decision about the factors that informed that outcome. This particular topic has been covered in the FastForward Labs report on Interpretability. For a simple single variable linear model, there is very little that needs to be explained, but the inner workings of neural networks or complex tree models are more challenging to explain and understand.

Techniques like Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP), which are discussed in the FFL report, provide good frameworks for understanding the reason a model made the decision that it did. However, explainability is not as straightforward to implement as auditability and reproducibility. While having explainable models is not always a regulatory requirement, it is often useful as a technique to check your own understanding of how the model works. In fact this was the case for the creators of LIME. Their original implementation of LIME highlighted how the model to which it was applied, while still reasonably accurate, was preferencing data attributes that the data scientists did not expect it to. This understanding helped them (the creators of the LIME) better understand why the model produced inaccurate predictions for a specific subset of the data it encountered.

While the churn application does need to be explainable to customers, explainability is very useful to the business users. Specifically, by understanding which variables inform the model predictions they are better able to decide which retention strategies might work best. The application within the churn example project provides a useful visualisation of the customer-related features that the model identifies as highly correlated[6] with churn.



The pneumonia model should be made as explainable as possible as it has real world consequences. The data is not tabular though, the model is just working with the input images. The project implements a version for LIME that highlights which parts of the image were considered important to the model when making the prediction. This should be something that is useful to someone with the required medical training to know if the model is looking at the right part of the image to make the predictions that it is.



Action Items:

- Decide if the model needs to have an explainability framework attached to it. If it is materially affecting someone's life, then it does.
- Choose an appropriate explainability framework for the model type and check if it makes the predictions more understandable to someone who is not involved in the project.
- Automate the process updating of this explanation framework with each model training job. If the model changes, it's likely that the explanation framework will become less effective for this new model.

About Cloudera
At Cloudera, we believe that data can make what is impossible today, possible tomorrow. We empower people to transform complex data into clear and actionable insights. Cloudera delivers an enterprise data cloud for any data, anywhere, from the Edge to AI. Powered by the relentless innovation of the open source community, Cloudera advances digital transformation for the world's largest enterprises.

Learn more at cloudera.com
US: +1 888 789 1488
Outside the US: +1 650 362 0488

Connect with Cloudera
About Cloudera:
cloudera.com/more/about.html

Read our VISION blog:
vision.cloudera.com

Follow us on Twitter:
twitter.com/cloudera

Visit us on Facebook:
facebook.com/cloudera

See us on YouTube:
youtube.com/user/clouderahadoop

Join the Cloudera Community:
community.cloudera.com

Read about our customers' successes:
cloudera.com/more/customers.html

Conclusion

Machine learning (ML) is an exciting discipline that provides opportunities for new insights and breakthroughs. However, in order for businesses to be able to justify their investment in ML, it is important that they are able to extract as much value as possible from their machine learning applications. In order for this to happen, it is important for ML models to progress beyond the proof-of-concept (PoC) phase and into production. This is the most effective way for the models' output to be integrated into a business's operations. Further, post-deployment, these models must continue to generate value for as long as they are 'in production'. Thus, it is really important that organizations' implementation of machine learning can deliver and support this type of sustained value. That's why COPML is such a useful framework for ML development – successful model development is the start, not the end of the ML lifecycle.

In this paper we have explored how COPML helps data science managers and their cross-functional teams make well-considered decisions about the tools, infrastructure and governance they need to build resilient and value-generating ML applications and systems. Getting these things right will also minimize the causes of friction in cross-team working by making it easier to collaborate effectively.

Selecting the right infrastructure enables an appropriate level of automation as well as a unified approach to security and governance. This increases the chance that your team(s) will be able to follow good practice without being overburdened by some of the more tedious requirements for achieving this. Wherever you are in your machine learning journey, COPML coupled with the Cloudera Data Platform can help you go further and deliver greater value.

Sources

1. Big Tech is used to collectively describe the most popular and best-performing American technology companies e.g. Facebook, Amazon, Microsoft, Apple, Netflix and Alphabet etc.
2. In practice, a model being trained for real world use would probably be trained on a dataset comprising a wider set of x-ray images e.g. x-ray images of lungs infected by diseases other than pneumonia.
3. This threshold will be determined by a number of factors such as the degree of variability in the underlying data, the volume of predictions made by the model relative to the number of examples it was trained on etc.
4. There are other risks as well. For example, the paper, The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks explores the risk of inadvertent memorization by neural network models.
5. https://gdpr-text.com/read/article-22/.
6. It is worth highlighting here that correlation isn't causation, consequently, there are risks to building retention purely on the basis of correlation. Cloudera Fast Forward Lab's Causality for Machine Learning report explores the application of causality in a machine learning context.

CLOUDERA

Cloudera, Inc. 5470 Great America Pkwy, Santa Clara, CA 95054 USA cloudera.com