**Module 8:**

# Deploying EO-based ML Models

Welcome to the ML4EO Module 8 on Deploying EO-based ML Models. This module heavily builds on the skills and practices presented in our previous modules. Here we will explore the concepts involved with deploying machine learning models that have been trained on earth observation data. Building models is just the first step; the true value of these models comes from their ability to analyze and interpret new data in real-world environments. Deployment moves models from the development stage to environments where they can provide actionable insights and solutions. This process is essential for harnessing the power of machine learning (ML) in the field of earth observation.

In **section 8.1**, we will discuss the journey of taking your ML models into prototyping and eventually production environments. We'll discuss the rationale for this transition, its significance, and the considerations that should be kept in mind to ensure a smooth and successful deployment. You'll learn about the concepts of real-world testing, scalability, integration, monitoring, and maintaining performance at scale.

**Section 8.2** introduces the concept of deployment frameworks and PaaS platforms that can facilitate rapid deployment of your models. These tools are especially useful in situations where development resources or expertise are limited, or when speed is a critical factor. PaaS platforms can simplify the deployment process, making it more accessible to a wider range of people and organizations. We'll explore several popular platforms and demonstrate how they can be used to deploy machine learning models trained on earth observation data.

By the end of this module, you'll have a comprehensive understanding of how to take your ML models, trained on earth observation data, from the development stage to real-world applications. You'll be equipped with the knowledge and tools to effectively deploy these models, driving meaningful impact in the field of earth observation.

**Learning objectives** - Participants will:

* Understand the concept of taking a trained and validated model beyond the academic cradle - and into a real world environment
* Are aware of various considerations and concepts involved with deployment (e.g. re-training, designing with the user in mind)
* Gain familiarity with tools and platforms for deployment
* Understand how to deploy a trained model to share with users

# 8.1 Taking ML models into prototyping and production environments

As a top-notch data scientist, you have sourced some really interesting geodata, cleaned and curated it, then trained a highly accurate machine learning (ML) model over a lengthy process of model selection and hyperparameter optimization. Now what? Migrating machine learning (ML) models from the lab enviornment into application prototyping and production environments is an essential part of the data value chain. In this section, we discuss the motivation behind deplpyment, the difference between prototyping and production, as well as the considerations and concepts involved with deployment of ML models.

**Motivation**

After all the considerable work we put into training ML models, it is important that we gain something from this work by sharing the model. It is the same whether in an open source or proprietary environment. Model deployment is the final objective in creating value from data as it allows us to **realize the potential for transforming data into actionable insights or automations.**

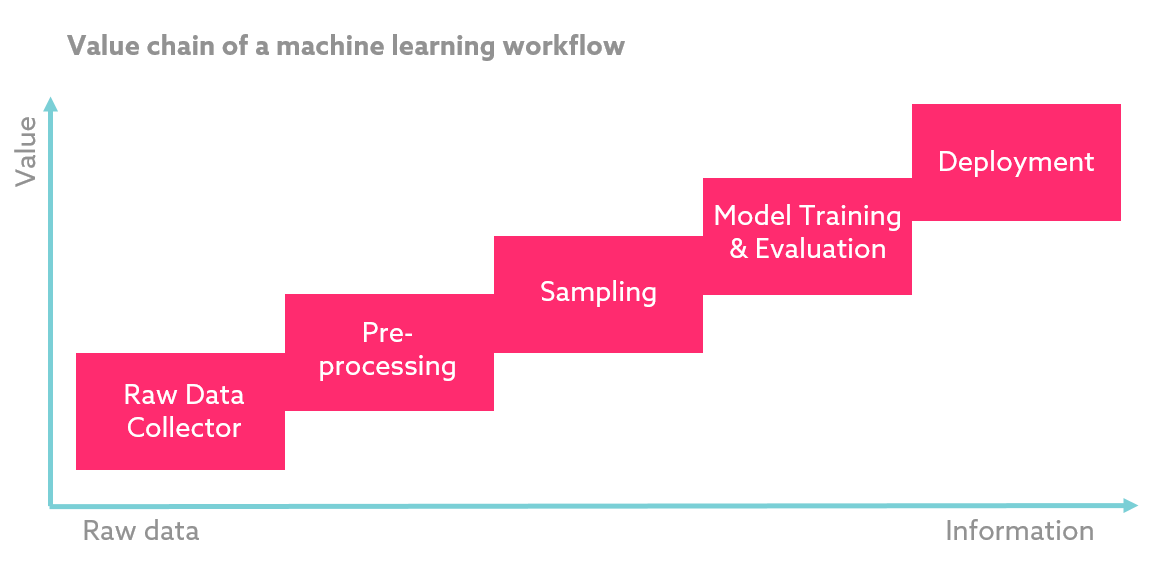
****

Figure 1: The data value chain in the ML context[[1]](#footnote-2)

The data value chain pictured in Figure illustrates the concept of adding value to data with each further step in the process. From raw data, we increasingly add value until the final point of model deployment. Looking at this in more detail, the key rationale for model deployment includes factors such as:

* **Application Integration:** The most critical reason for moving ML models into production is to integrate them with existing systems and applications. In many cases, these models are not used in isolation; rather, they are part of a larger ecosystem of applications and services that rely on these predictions to function effectively.
* **Real-world Testing:** Prototyping and production environments provide real-world data, which may not be fully reflected in the training data. It helps in understanding how the model performs with live data, and it's necessary to validate the model's accuracy, reliability, and robustness in these real-world situations.
* **User Feedback and Iteration:** Prototyping and production environments allow for end-user interaction. Feedback from users is key in the development of any application, as it helps in iterative refinement and can guide the development of new features. Real user interaction can reveal issues and opportunities that might not have been apparent in the development stage.
* **Scaling:** Models that work well with a small dataset may struggle when the volume of data increases in a real-world environment. It's important to test how well your infrastructure can handle increased data loads and how the model behaves with that scale of data.
* **Continuous Learning (retraining):** Some ML models can benefit from continuous learning, where they refine their predictions over time based on new data they receive. This is only possible in a production environment where the model can interact with real-world data in a live setting.
* **Regulatory and Ethical Compliance:** Deploying the ML model in a controlled environment allows for monitoring and managing issues related to privacy, fairness, and regulatory compliance, which are often overlooked during the development stage.
* **Business Value:** Ultimately, the goal of developing ML models is to derive business value, such as cost savings, increased efficiency, or improved decision-making. This is only possible when the models are applied to real-world problems in production environments.

In an open source context, we may choose to release the model as a publicly-accessible application on various web-based platforms. In proprietary contexts, we may use more secure platforms and limit access to particular users. In both cases, the modality is the same – i.e. we deploy the model as an application that can add value to users and/or add value to further development of the model/application by providing feedback to the development team.

**Deployment Modalities: Protyping vs Production**

During university, you have likely trained models and shared the results (and possibly model checkpoints) with your professor or cohort. You may even have developed an application from the model and deployed it for prototyping purposes. If you have worked in industry, you may also have deployed models at the production-level applications. The objective of this module is ensure that you **understand how to deploy a trained model to simple prototyping environments** so that you can share a basic application with users for feedback purposes. However, we will also discuss production deployment in brief so that you have some familiarity with the terms and concepts. We will therefore first define the differences between prototyping and production.

Deploying ML models for prototyping vs. production involves different considerations and requirements. The choice depends on your end goal: **prototyping is generally used to evaluate feasibility and design**, while **production deployment is aimed at supporting real-world, operational applications**.

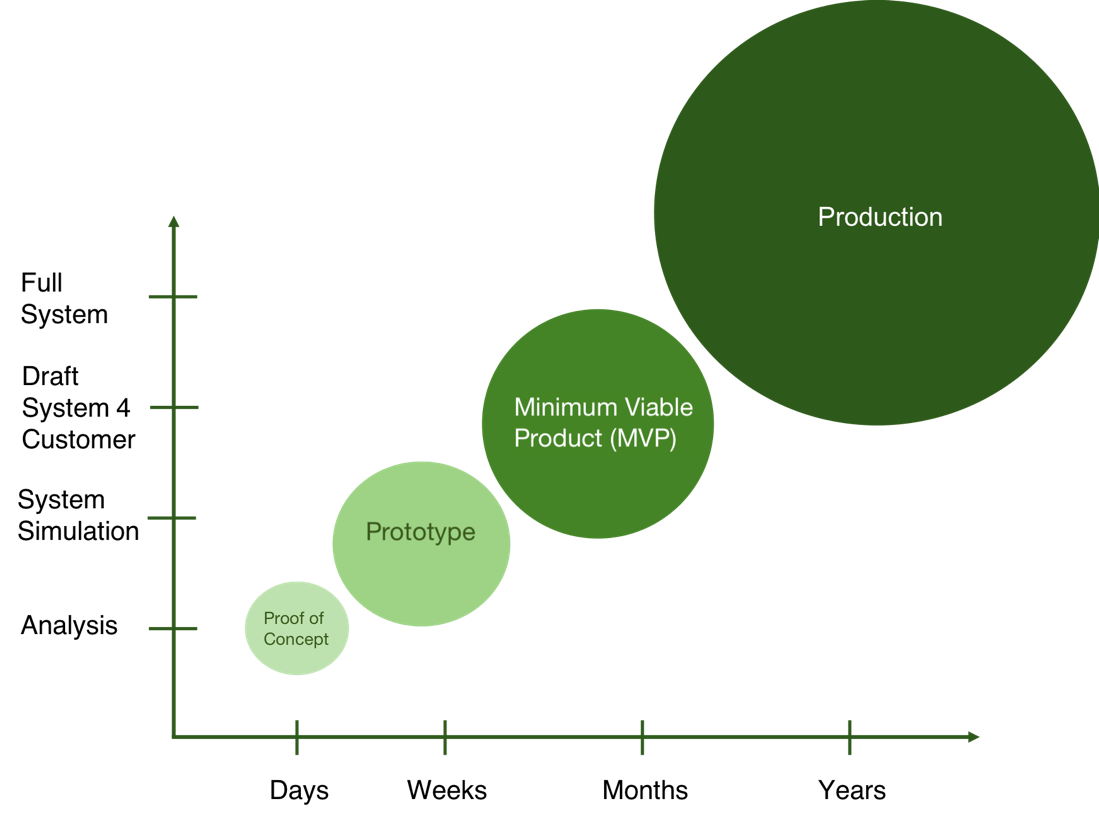


Figure 2: Product development stages showing POC/Prototyping vs. MVP/Production[[2]](#footnote-3)

Figure 2 shows the relationship between the various stages of product development. The concept is a software development paradigm, rather than something tailored for ML. However, the same relationship holds true – i.e. the various stages involve exponentially increasing levels of input and corresponding complexity. In our context, we can think of Proof of Concept (POC) as the stage at which you have a working model ready to go. The next stage is to deploy it for prototyping purposes. Once you have some solid feedback from your prototype, you may want to deploy your application as an MVP, releasing to a wider user base. The distinction between this stage and actual ‘production’ is often blurred, but in general you can think of production as the more mature and stable stage beyond MVP, where your model-based application is established and serving value to a large user base.

In short: prototyping is simple; production is usually more complex. The key differences are as follows:

* **Scale:** Prototyping deployments typically deal with small-scale data and users, often in a controlled or simplified environment. Production deployments, on the other hand, are designed to handle large-scale data and multiple simultaneous users in a more complex, real-world environment.
* **Robustness:** In production environments, ML models need to be highly robust and reliable, handling a variety of inputs and conditions without failing or producing erroneous outputs. Prototyping deployments may allow for more flexibility and tolerance of errors as the model is refined and improved.
* **Infrastructure:** The infrastructure for a prototype may be relatively simple, often running on a single machine or server, or a managed, PaaS platform (ref. 8.2). A production deployment usually requires a more complex infrastructure, including potentially distributed systems or cloud services, to handle high-volume traffic, data storage, security, scalability, and redundancy needs.
* **Maintenance and Monitoring:** In a production environment, continuous monitoring and maintenance are required to ensure the model's performance over time. This involves ongoing evaluation of the model's metrics, troubleshooting issues, and potentially updating or retraining the model. Prototypes, by contrast, often don't require the same level of ongoing support.
* **Integration:** Production deployments typically require seamless integration with other systems or applications, often through APIs or microservices. Prototyping deployments are usually standalone, focused on demonstrating the feasibility or effectiveness of the model.
* **Data Management:** In a production environment, data pipelines need to be established for real-time or batch data processing. This involves dealing with real-world data quality issues, missing values, or changes over time. Prototyping might involve a static, pre-processed dataset, mock data, or restricted types/volumes of data to be provided by the users for inference.
* **Security and Privacy:** These are especially important in production environments where real user data is involved. Prototyping environments may use mock or anonymized data, reducing privacy concerns.
* **Testing:** In a production environment, rigorous testing is critical before deployment, including unit tests, integration tests, stress tests, etc. Prototyping might involve less extensive testing.

**Deployment Considerations**

Taking our trained ML models into prototyping or production environments involves a variety of important considerations to ensure optimal performance. In general, our discussion in this section will apply more to production deployments, but some concepts are applicable for prototyping as well. We discuss the key points below.

**Infrastructure and Tools**

Prior to deployment it is important to consider which infrastructure and tools will be necessary to deploy the model including hardware, cloud services, and ML platforms or frameworks. The infrastructure should be robust, reliable, and secure – and informed by the other key points mentioned in this subsection. Depending on your exact role in the organization, you may also consider the costs associated with the compute resources required to train and deploy the model, storage, and personnel.

Infrastructure is usually the domain of a data engineer or IT infrastucture specialist, depending on your context. If you are prototyping, or working in a resource-constrained context, then you will likely seek a managed infrastructure solution such as a PaaS platform like Heroku (ref. 8.2). These platforms typically offer different levels of service to meet the needs of your application.

**Scalability**

Scalability refers to the capacity of a system, network, or process to handle a growing amount of work or its potential to expand to accommodate that growth. In the context of ML model deployment, scalability can significantly impact the efficiency and reliability of the deployed models. From a data scientist’s perspective, the key items to note with regard to scalability are as follows:

* **Data:** The deployment model should be able to handle ingest larger amounts of data. In computer vision generally, and particularly with EO data applications, data scalability is a major concern depending on the specifics of the applications. This could mean higher user volumes, or increased requests per user, or simply massive amounts of data (e.g. where a time series is used for inference). Another example of this could occur in the future when higher-resolution geodata becomes more widely accessible. Will your model be able to handle this? Speaking with data engineers and IT infrastructure specialists can help in planning for such scenarios.
* **Model:** Larger or more complex models (e.g., deep learning models) can be more computationally intensive, both to train and to use for making predictions. The deployment infrastructure must be capable of supporting these models in a timely and efficient manner. In general, the financial costs of the required IT infrastrcture will increase along with increased compute and decreased serving time. For a prototype, it may be possible to utilize a bare-bones deployment using minimal resources, as the the objective is simply to get feedback on inference. In a production environment, users will expect a much higher level of service. In such cases, the trade-off between model architecture, size and deployment resources is best discussed with your data engineering team.
* **Inference:** This refers to the system's ability to serve predictions. When the model is deployed in a production environment, it may need to handle many simultaneous prediction requests. The system must be designed to accommodate this computational load, possibly by using techniques like load balancing or horizontal scaling (adding more machines to the serving cluster). Again, data engineers and infrastructure experts should be consulted.
* **Retraining:** If the model needs to be retrained regularly (e.g. online learning), the system must be able to handle the computational demands of regular retraining. This might involve distributed computing or the use of specialized hardware like GPUs.

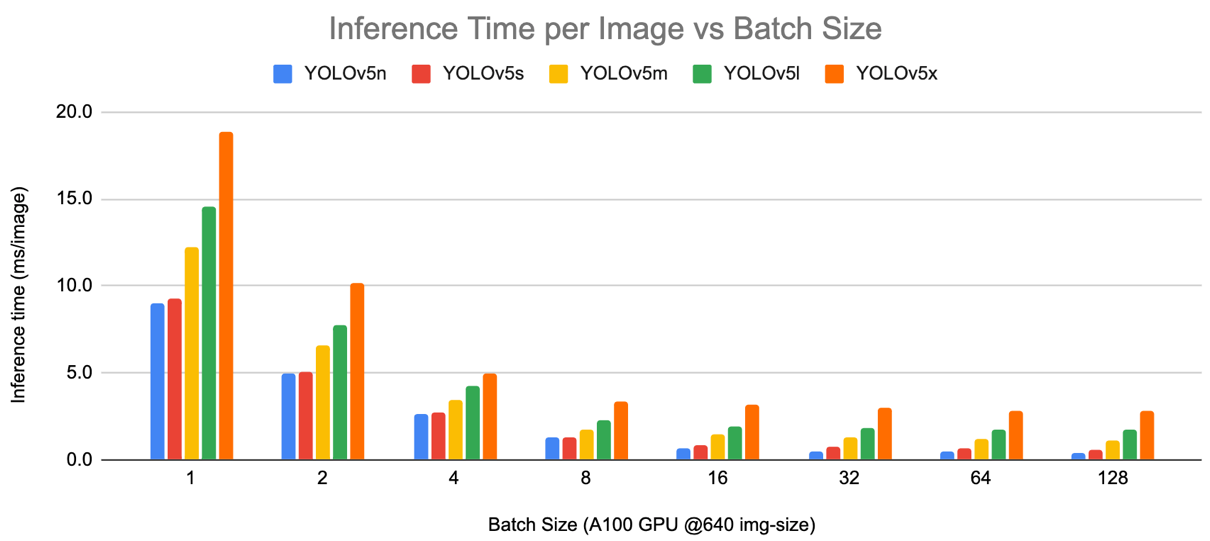


Figure 3: Effects of model inference time on data ingestion (batch size)[[3]](#footnote-4)

Figure 3 demonstrates the relationship between batch size and inference time for a popular computer vision model. You may already have a background in computer science or data engineering, but even if not, you can probably understand intuitively how batching data results in faster processing. For instance, in most endeavors, doing a repetitive task a fixed number of times is more efficient than doing the same number of tasks, interspersed with other, different tasks because of the overhead involved with task switching. We might therefore naturally suggest a high batch size when liaising with our data engineering team. However, there is a tradeoff from the user experience perspective as increased batch sizes may lead to longer wait times -depending on the context. It is important to consider all such factors when making these types of decisions.

Scalability planning should be done in the early stages of model deployment planning to prevent system overloads and ensure smooth operation as data, inference demands, and the complexity of models increase. Though as a data scientist you will not likely have to configure actual IT infrastructure, it is important to maintain an open dialogue with data engineers to ensure the deployment requirements are clearly communicated.

**Monitoring and Updating**

Monitoring and updating are processes that allow for the continual assessment and improvement of the model's performance over time, ensuring its ongoing reliability and accuracy. **Monitoring** involves tracking the performance of your ML model over time and detecting any changes or anomalies. Monitoring commonly comprises the following components:

* **Performance Metrics:** Continuously track metrics like accuracy, precision, recall, F1 score, or any other relevant metrics for your model. This helps in understanding whether the model is still performing well or if its performance is degrading.
* **Logging and Alerting:** Logs record model predictions (performance) and system behavior. This allows for easier debugging if issues arise. Additionally, implementing alerts that notify you when key metrics fall below a certain threshold can be very useful.
* **Drift Monitoring:** The term **data drift** refers to a change in the input data's distribution over time, while concept drift refers to a breakdown in the relationship between the explanatory and response data used to train a model in the first place. Both can significantly impact the performance of deployed models over time. Data and concept drift can have various causes including natural evolution of data, unexpected external events (e.g. COVID-19), changes in data collection process (e.g. improved satellite imaging sensors), and seasonality that was not well-represented in the training data. Monitoring for drift can give early indications of potential impacts to performance.

A picture containing text, diagram, screenshot, circle

Description automatically generated

Figure 4: Data drift: changes in the distribution of model explanatory variables over time[[4]](#footnote-5)

Figure 4 illustrates two examples of data drift. The left hand plot shows the original training data and decision boundary. The plot on the top right shows a case where the data distribution has drifted, but the classifier still predicts correctly, while in the bottom plot the data distribution has also drifted, but the classification is incorrect. Figure 5 below provides an example of concept drift. In this case, the distribution of the explanatory variables has not changed, but the real-world relationship with the response variable has. Therefore, one of the data points has been incorrectly classified by the out-of-date classifer.

A picture containing diagram, screenshot, line, circle

Description automatically generated

Figure 5: Concept drift: changes in the real-world relationship between explanatory and response variables are not reflected by the model

**Updating** is a consequence of monitoring. Depending on what the monitoring shows, there may be a need to update the model to ensure it continues to perform well. Updating could involve:

* **Retraining:** This involves retraining the model on new data, especially when there has been significant drift in the incoming data.
* **Model Tuning:** If the model's performance is degrading, it might be necessary to tune the model's parameters or even consider a different algorithm.
* **Feature Engineering:** You might need to revisit the features being used by the model, especially if the relationship between the features and the target variable has changed over time.
* **Model Versioning:** When updates are made, it's crucial to manage different versions of the model. This allows for easy rollback to a previous version if something goes wrong and helps maintain a history of changes.

**Testing**

Testing involves ensuring the model works as expected before and after it's deployed in the production environment. There are several aspects to consider when planning for model testing. Some of these are valid in the prototying environment, while others are more production orientated. You may recognize the terms from software development practices. Indeed, many production deployments necessitate the involvement of a software engineer to develop a presentable user interface and functional backend.

* **Unit Testing:** This involves testing individual components of your code to ensure they work as intended. In the context of machine learning, this might include testing data preprocessing steps, model training code, or custom metrics.
* **Integration Testing:** This involves testing the system as a whole to ensure different components work well together. This might include testing the entire pipeline from data ingestion, preprocessing, inference, and results storage.
* **Load Testing:** This involves testing the model under heavy loads to ensure it can handle the expected number of requests in production. This is especially important for systems where predictions need to be served in real time.
* **Adversarial Testing:** This involves testing an application with edge case inputs to ensure it doesn’t break. In the context of ML, this could include inputs that are unusual, outside of the distribution of the training data, or designed to try to trick the model.
* **A/B Testing:** This involves running two versions of the application side by side on a subset of traffic to compare some aspect of the application or performance. This can be helpful to assess whether a new application feature is an improvement over the old one before deploying the new feature to all users. In the context of ML, this could include comparing two versions of a model in terms of the impact on the users. For instance, a company providing advice to farmers could offer a free platform for farmers to predict crop yield over the coming year by entering the coordinates of their field. Two models could be tested on different randomly assigned groups, with the users providing feedback on accuracy over the year.

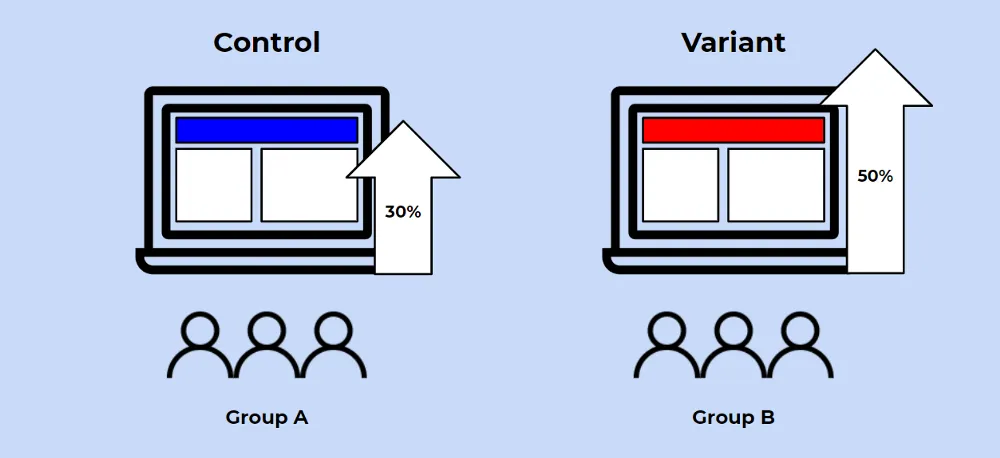


Figure 6: A/B Testing a new application feature with a control group and a test group[[5]](#footnote-6)

**Data Privacy and Security**

Depending on the context, there may be legal requirements you have to meet when processing data in an ML-based application. In the context of large EO training datasets, we don’t often think of data privacy as applicable. However, when deploying a model in a production environment and accepting input data from users as input, it is important to consider all possibilities.

According to the General Data Protection Regulation (GDPR) of the European Union, "personal data" is defined as any information relating to an identified or identifiable natural person, also known as a "data subject". An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, **location data,** an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person. Many countries are adopting similar legislation. For instance, on 13 October 2021 Rwanda enacted its own data protection legislation (Law No 058/2021 Relating to the Protection of Personal Data and Privacy)[[6]](#footnote-7).

Referring to the example presented in the section on A/B Testing (above), the data entered by the farmers providing the coordinates of their field would certainly qualify as “personal data” according to the GDRP definition alone. Therefore it is very extremely important to consider provisions for compliance with applicable data privacy and security regulations, as well as the potential legal implications in cases of non-compliance. When planning deployment, it is best to coordinate with your organization’s Data Protection Officer (DPO), Chief Data Officer (CDO) or legal counsel if available.

# 8.2 Frameworks and Platforms for Rapid Deployment

In this subsection, we will explore several established deployment frameworks and platforms as a service (PaaS) that simplify the process of deploying ML models. Each of these cloud-based platforms offer unique features and capabilities, with various strengths and weaknesses. They can significantly streamline the deployment process, making it easier to put ML models into production. However, the best choice depends on your specific context, such as the scale of deployment, budget, security requirements, and the team's technical expertise. We will discuss the features of each platform, to help familiarize you with popular market offerings to meet your ML deployment needs.

**Motivation**

Assuming that you have your model trained and ready to go, and you have some idea what you want the application to do and look like. Do you have a data engineering team that can setup a Databricks cluster to power the app backend, and a software development team to create a high quality user interface with input from a UX/UI specialist? In the case of many organizations, the answer is no. You may be developing your own startup and have an extremely limited budget. You may simply want to get a very rough prototype up and running so that you can get user feedback with the minimal investment. Purpose-built deployment frameworks and PaaS offload the IT infrastructure overhead from the user. This provides a means for getting a model deployed and running with the least amount of effort and resources.

In the context of ML model deployment, PaaS platforms serve two primary use cases:

1. **Prototyping**: As they are incredibly easy to use, PaaS platforms allow data scientists to quickly deploy models and applications with minimal setup – i.e. to focus more on the modelling and less on the intricacies of deployment, thereby accelerating the prototype development process. In some cases, this can involve a simple git commit to deploy a code base and model to an application. The service handles everything else. This works perfectly for prototyping, where the goal is not to release a finished product, but to provide an early version for testing with users in order to gain feedback.
2. **Production Deployment for Low-Resource Contexts:** Along with being easy to use, PaaS platforms can be relatively cheap for simple use cases depending on the resources required. At the same time, they can be configured to provide highly professional looking applications, and backend compute resources scaled to ensure adequate user experience. This makes them ideal for organizations with limited IT capacity.

This subsection will particularly focus on the use of frameworks and PaaS in the prototyping context. However, all platforms discussed are capable of deploying high quality productions services as well.

The choice of whether to use these platforms depends on your specific context. Below we summarize the general strenghts and weaknesses:

**Strengths**

* **Cost:** Can be cheap or free for very small scale applications such as prototyping.
* **Ease of Use:** Often provide a user-friendly user interface and straightforward procedures for deploying applications, reducing the need for specialized knowledge.
* **Integration:** Typically offer seamless integration with popular version control systems like Git and can easily connect with other cloud services.
* **Scalability:** Designed to scale with your application. They can automatically manage resources based on traffic, which can be a huge advantage for ML models that may need to handle large volumes of requests for data ingestion (think raster data time series!).
* **Managed Services:** Often come with managed services like databases, caching, and queueing systems, which can be easily integrated into your application.
* **Abstraction:** Abstract away the server and infrastructure management, allowing developers to focus on the application logic and model development instead of server setup and maintenance.

**Weaknesses**

* **Cost:** Cost can be a weakness, depending on your use case. While these platforms often come with a free tier, scaling up the resources and utilizing additional features can quickly become costly.
* **Limited Customization:** Take control away from the developer in order to provide simplicity and ease of use. This can be a limitation when you need to do something that doesn't fit into their standard deployment process.
* **Vendor Lock-in:** Once you develop and deploy an application on these platforms, it can be difficult to move it to another platform or to an in-house server due to their specific set of tools and services. This depends heavily on the use case, as with simple applications it can be quite easy to simply migrate your code to another provider.
* **Cold Start Problem:** In some cases, such as with Heroku's free tier, applications can "go to sleep" after a period of inactivity and may take some time to "wake up" and start serving requests again.
* **Data Security and Compliance:** Depending on the nature of your application, there may be data security and compliance issues to consider, as your data is stored on third-party servers.

**Background on Cloud Services**

While an in-depth discussion of cloud services is beyond the scope of this course, we provide here a brief explanation of PaaS to help you understand the concept. You likely have some familarity with some of these concepts already. SaaS stands for software as a service, while IaaS stands for infrastructure as a service. IaaS is the base level of cloud services, where you can rent virtual machines (VMs) and other virtual services from cloud service providers that make these services available using real machines in real data centres. PaaS is one level of abstraction up from IaaS. Figure 7 below provides some background of where PaaS fits with the rest of the cloud computing services stack, along with common services at each level. It should be obvious that as we move up in the stack, we have correspondingly less control over the underlying software and hardware. However, this comes also at a proportional decrease in complexity. Such is the magic of cloud services - where an end user that once had to buy and maintain expensive software running on a local machine, can now simply subscribe to a reliable SaaS that is entirely managed by the software provider. As data scientists, we benefit from many SaaS, but also from PaaS. Some of use may also work at the IaaS level, depending on our organizational context.

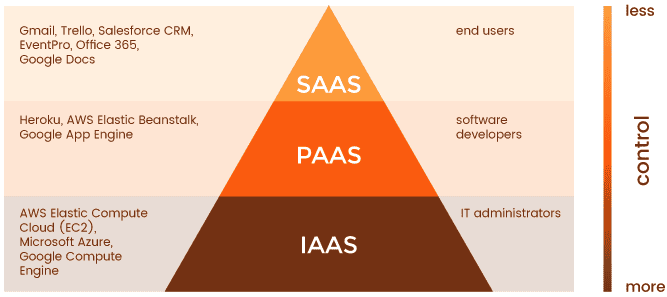


Figure 7: PaaS in the larger context of cloud services

**Comparison of Popular Deployment Platforms**

Here we discuss the distinguishing features of various popular deployment frameworks and PaaS platforms for application deployment. These platforms, including Heroku, Streamlit, Anvil, and Google Cloud Platform's Vertex AI, are renowned for their capacity to significantly simplify and accelerate the deployment of applications, including those based on ML models. Each platform offers a unique blend of functionalities, from smooth integration with other services to in-built machine learning capabilities and auto-scalability. The following exploration will provide you with an introduction to each platform, which will help you understand what type of services are available on the market.

[**Heroku**](https://www.heroku.com/)

Heroku is a cloud-based, Platform-as-a-Service (PaaS) that developers use to build, run, and operate applications. It's particularly well-suited for deploying simple applications based on ML models due to its ease of use, scalability, and variety of supported languages and tools.

**Features:**

* **Multiple Language Support:** Heroku supports several popular programming languages, such as Python, Java, Ruby, Node.js, PHP, and Go. This means you can build your machine learning models in the language you're most comfortable with.
* **Easy Deployment:** Deployment of applications is straightforward with Heroku. You can deploy directly from a Git repository, and setting up a continuous integration/continuous deployment pipeline is relatively simple.
* **Add-Ons and Services:** Heroku provides a plethora of add-ons (data stores, monitoring services, etc.) and fully-managed data services like Postgres, Redis, and Apache Kafka.
* **Scalability:** Heroku makes it easy to scale your application. You can adjust resources allocated to your application depending on your needs.
* **Heroku CLI and Dashboard:** Heroku provides a command line interface (CLI) and a user-friendly dashboard for managing your applications, resources, and add-ons.

Heroku was founded in 2007 as a platform for Ruby developers. Salesforce, a cloud-based software company, acquired Heroku in 2010. Over time, the platform expanded its language support to include Java, Node.js, Scala, Python, PHP, and Go, becoming a popular choice for application deployment.

Heroku's unique selling proposition (USP) is its simplicity and ease of use. It abstracts away the complexities associated with server and infrastructure management, letting developers focus on the application code. It is particularly well-suited to developers and small teams who want to deploy code quickly without worrying about infrastructure setup. With its robust ecosystem of add-ons and services, Heroku also simplifies tasks like monitoring, scaling, and database management, making it a comprehensive platform for deploying applications, including machine learning models.

Heroku used to offer a free plan, which made it one of the most popular platforms for deployment. Unfortunately this stopped in November of 2022. However, the platform still offers a very affordable tier for prototyping that costs between USD 5 and 7 per month.

We **will be using Heroku for the practical exercises in this module to deploy** an EO data-based ML model application.

[**Streamlit**](https://streamlit.io/)

Streamlit is an open-source Python framework and managed platform that makes it easy to create and share beautiful, custom web apps for machine learning and data science. Developers and data scientists can quickly turn their Python scripts into interactive web applications without the need for web development skills.

**Features:**

* **Simplicity:** Streamlit allows you to turn Python scripts into data apps very easily and quickly. It reduces the gap between prototyping a machine learning model and sharing it as a web application.
* **Interactive Widgets:** Streamlit supports interactive widgets like sliders, buttons, and inputs that you can use to interact with your data and machine learning models.
* **Data Compatibility:** Streamlit is compatible with major data science libraries, including Pandas, NumPy, Matplotlib, and others.
* **Real-Time Interaction:** Streamlit's applications offer real-time interactivity, and the user interface refreshes instantly as the user interacts with it.

Streamlit was founded in 2018 with the aim of simplifying the process of creating and sharing ML tools. It quickly gained popularity in the data science community because of its simplicity and the speed with which you can create apps. Streamlit has become a go-to tool for data scientists to quickly prototype and share their models as web applications.

Streamlit's unique selling proposition (USP) is its focus on simplicity and speed. It allows data scientists to quickly turn Python scripts into interactive web applications without requiring knowledge of web development. With just a few lines of code, users can build interactive, real-time applications that can showcase machine learning models or data exploration tools. This makes it ideal for prototyping and for the rapid iteration and development of ML model applications. Streamlit fills a niche for data scientists and developers who want to focus on the data science and ML aspects of an application, while minimizing the time and effort spent on web development.

Streamlit itself is a free and open-source Python library that you can use to create data applications. You can run these applications locally or deploy them on your own server for free. The Streamlit hosted platform offers a free tier that allows you to deploy and manage your apps. The free tier includes a single container for app deployment.

[**Anvil**](https://anvil.works/)

Anvil is a comprehensive platform for building full-stack web applications with nothing but Python. It's a modern tool for deploying applications, including those based on machine learning models, and is excellent for Python developers who want to build web apps without having to learn HTML, CSS, and JavaScript.

**Features:**

* **Full Python:** Anvil is a full-stack web app framework written in Python. It lets you write Python on the front end, the back end, and even in your database queries.
* **Drag-and-Drop Editor:** Anvil has a drag-and-drop UI editor that enables you to design the appearance of your application without needing to write any HTML or CSS.
* **Serverless Deployment:** Anvil handles the hosting and deployment of applications, so you don't have to worry about server maintenance or setting up a hosting environment.
* **Version Control:** Anvil supports Git integration, allowing you to work on your application collaboratively with version control.

Anvil was founded with the mission to fix web development by bringing it into the modern era. It started as a full-stack web framework that lets you write Python everywhere, which appealed to developers who were proficient in Python but didn't want to switch languages to build web applications.

Anvil's unique selling proposition (USP) is its full-stack Python environment. It allows developers to build web apps using just Python, with a drag-and-drop interface that eliminates the need for HTML, CSS, or JavaScript. This makes Anvil particularly appealing to data scientists and ML engineers, who often work in Python and want to create web applications for their models without having to learn a new language.

Anvil offers a free tier with access to many of its core features, including unlimited apps, hosting and local deployment. However, some advanced features and increased capacity are only available on paid plans.

[Vertex AI](https://cloud.google.com/vertex-ai)

Vertex AI is a managed ML platform from Google Cloud that allows developers and data scientists to build, deploy, and scale ML models quickly and efficiently. It unifies the Google Cloud's ML offerings under one brand and introduces new tools to improve and streamline ML workflows.

**Features:**

* **Unified API:** Vertex AI offers a unified API, client library, and user interface to ease the development of ML models.
* **AutoML:** Vertex AI includes AutoML capabilities that allow users to train high-quality models with minimal effort and ML expertise.
* **Custom Model Support:** Vertex AI allows the training and serving of custom models using frameworks such as TensorFlow, PyTorch, and Scikit-learn.
* **Scalability:** Vertex AI handles the provisioning of resources, making it easy to scale the deployment of models depending on the demand.
* **MLOps Features:** It also provides MLOps (Machine Learning Operations) tools to manage the ML lifecycle, including model versioning, A/B testing, and a continuous evaluation service.

Google Cloud launched Vertex AI in 2021 to provide a unified platform for ML and AI. It incorporates the functionality of Google Cloud's existing AI and ML services like AutoML and AI Platform, while introducing new tools for managing the ML lifecycle.

The USP of Vertex AI is the unification and simplification of the ML workflow in a single, easy-to-use managed service. It offers both AutoML and custom model support, meaning it caters to ML practitioners of all skill levels. Additionally, it's integrated with Google Cloud's robust infrastructure and security, ensuring that models are served reliably and securely.

Google Cloud Platform does offer a free tier that includes some resources and tools within Vertex AI. However, beyond those free resources, charges are based on usage. For the most accurate and up-to-date information, refer to the pricing page on the official Google Cloud website.

1. Image source: https://blog.solvatio.com/en/using-machine-learning-to-analyse-data-and-how-it-can-help-businesses [↑](#footnote-ref-2)
2. Source: <https://cloudflex.team/blog/mvp-vs-poc-vs-prototype/> [↑](#footnote-ref-3)
3. https://github.com/ultralytics/yolov5/discussions/6649 [↑](#footnote-ref-4)
4. Source: https://towardsdatascience.com/dont-let-your-model-s-quality-drift-away-53d2f7899c09 [↑](#footnote-ref-5)
5. Source: <https://towardsdatascience.com/a-simple-guide-to-a-b-testing-for-data-science-73d08bdd0076> [↑](#footnote-ref-6)
6. [The new Rwandan data protection law | ALB Article (africanlawbusiness.com)](https://www.africanlawbusiness.com/news/17478-the-new-rwandan-data-protection-law) [↑](#footnote-ref-7)