TOOL

Deploying, Hosting, and Monitoring Your Model

Introduction

Once you have trained and tested your machine learning model, you have to deploy and host your model in a production environment so as to make your model available to stakeholders to help solve their business problems. Deploying a machine learning model can be challenging, but advancements in cloud computing have made the process much easier.

This reference tool outlines the considerations for deploying, hosting, and monitoring machine learning models to ensure their ongoing success. Use it as a guide when you are prepared to deploy your machine learning model.

1. Determine the deployment method.

After the model is fully trained, it can be deployed to make predictions on new data through two primary methods: batch inference (or offline inference) and online inference (also known as real time). Batch deployment processes large volumes of data in batches on a recurring schedule and stores the predictions in a database. This information can be provided to stakeholders when needed. Online inference processes data as it arrives in real time.

The choice between the two methods depends on the specific problem requirements, and both come with their own set of advantages and disadvantages.

Deployment Method	Pros	Cons
Batch inference	• Can deploy more complex models with a larger number of inputs • Requires simpler infrastructure requirements and less computational power compared to online inference • Predictions can be analyzed and processed before being seen by stakeholders	• Can result in higher processing latency • Not suitable for applications that require real-time predictions, therefore predictions may not be available for new data
Online inference	• Provides results in real time and on demand. • Lower processing latency	• Harder to implement • Cannot handle as complex models as batch inference • More computationally demanding compared to batch inference

Key considerations:

- Your latency requirements
- The complexity of your model
- Specific requirements of your use case

Questions to ask:

- How frequently do you require predictions?
- Do you need results based on individual cases or batches of data?
- What amount of computational power is needed to process the inputs?

2. Choose a hosting environment.

When deploying a machine learning model, one important consideration is where to host it. There are two main options — internal hosting or cloud services — and the decision often depends on the specific needs of the project and organization. Below are some key considerations when choosing where to host your model.

Hosting option	Pros	Cons
Internal	• Greater control and security for sensitive data • May be more cost effective for larger companies with existing internal infrastructure	• Can be more costly in resources, time, and maintenance • More difficult to scale
Cloud services	• Easily scalable and flexible, with lower maintenance • More cost effective for smaller companies without existing internal infrastructure	• May not be cost effective on large projects • May not meet strict security requirements

Questions to ask:

- **Infrastructure:** Does your organization have the necessary hardware, network, and security protocols to support internal hosting? If not, consider cloud services.
- **Cost:** How much will it cost to host and deploy the model? Internal hosting may require a significant upfront investment, while web services are often charged based on usage.
- **Scalability:** Will your project grow in terms of data volume and user demand? Choose a host that can easily scale up to meet those needs, such as cloud-based web services.
- **Security:** Does your project have specific, strict security requirements? If so, deploying the model on-premises can provide better control over the infrastructure and data.

Popular cloud services for deploying and hosting machine learning models

Deploying ML models to the cloud is an increasingly common practice, as it provides easy access to necessary computing power, storage, and network resources for handling large data volumes and running complex models. Yet there are also potential downsides to consider when weighing your options, including security concerns, high cost, latency, and dependency on the cloud provider.

- **Amazon SageMaker** is a fully managed machine learning service offered by Amazon Web Services (AWS). It provides a complete platform for building, training, and deploying machine learning models.
- **Google Cloud AI Platform** is a suite of tools and services that help you build, train, and deploy machine learning models on Google Cloud.
- **Microsoft Azure Machine Learning** is a cloud-based machine learning platform that enables you to build, deploy, and manage machine learning models.
- **BentoML** is an open-source platform for deploying, managing, and serving machine learning models. It provides a unified interface for packaging and deploying models as production-ready web services.
- **Kubeflow** is an open-source platform for deploying and managing machine learning workflows on Kubernetes. It provides a unified interface for managing machine learning pipelines.
- **TensorFlow Serving** is a framework for serving machine learning models using TensorFlow. It provides a flexible architecture for deploying models in production.

3. Deploy your model.

Once you've chosen the deployment method and hosting environment that suits your project, the next step is to package the model along with its dependencies into a deployable format such as a container or a bundle. **Containers** are popular because they're predictable, reproducible, and easily modifiable, making them ideal for collaboration among engineers.

Industry-standard tools that specialize in preparing for deployment:

- **Docker** is a popular tool for packaging and deploying applications in containers. It can be used to package machine learning models and their dependencies into a container that can be easily deployed to different environments.
- **Kubernetes** is an open-source container orchestration platform that can be used to manage and scale containerized applications. It can be used to deploy and manage machine learning models packaged in containers.
- **ONNX Runtime** is an open-source runtime engine for deploying models that are compliant with the Open Neural Network Exchange (ONNX) format. It provides high-performance execution of models across different hardware platforms.

Before deploying the packaged model, you should test it to ensure that it performs as expected. This may involve running tests to evaluate the model's accuracy, precision, recall, and other performance metrics. Once you are satisfied with the results of your testing, deploy the packaged model to your target environment that has been determined based on your security, financial, performance, and computational requirements.

Automating deployment

To streamline deployment and testing workflows, some organizations automate the process. Automation ensures the model is tested regularly to maintain its robustness. It can also help scale the model without burdening the team.

4. Monitor for model improvements.

Monitoring a deployed machine learning model is vital to ensure it is performing well and to detect any issues that may arise during production. Monitoring the model's performance, identifying errors, and making necessary adjustments is crucial. Things to monitor for include:

- **Performance deterioration:** Quality degrades over time due to changes in data distribution.
- **Bias or discrimination:** This occurs when the data used to train the model is not representative of the population it is intended to serve.
- **Security risks:** Over time, attackers may be able to manipulate or alter the model's predictions to achieve their goals.
- **Costly revisions:** A model may require costly revisions or even replacement if its performance degrades over time.

Best practices for monitoring model performance

- Constantly evaluate your ML model's performance on real-world data to detect any decrease in accuracy due to changes in the data environment, known as model drift. If model drift occurs, retrain your model with fresh data to improve its accuracy.
- Monitor your deployment pipeline to ensure the model runs smoothly and debug it when necessary.
- Collect feedback from end users to improve the model's performance by identifying areas for improvement and providing valuable insights.

Considering all of the above factors and being methodical when deploying a machine learning model is important to ensure that the model is accurate, reliable, and delivers the expected value to the business or project for which it is intended.