Azure Machine Learning service vs Azure Machine Learning Studio

Because the names are similar, it's important to distinguish between Microsoft Azure Machine Learning Studio and the Azure Machine Learning service. Both of these products can be used to develop and deploy machine learning models, but they support different needs.

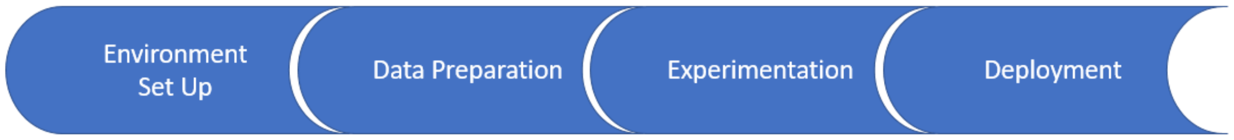
Azure Machine Learning Studio is a drag-and-drop visual workspace that is user-friendly and doesn't require any programming. It's ideal for learning data science and for small machine learning projects.

Azure Machine Learning service provides callable services to scale, monitor, and deploy machine learning models. It's meant to augment and extend the data science pipeline with integrated Azure services. The Azure Machine Learning service SDK lets you write code that uses these prebuilt services and integrate them into your data science workflow.

You start your model development on your local computer with the Python tools and open-source frameworks of your choice. Then use the Azure Machine Learning service via Python modules to speed up model training and evaluation, hyperparameter tuning, and deployment, complete with a callable REST API.

Azure Machine Learning Service within a data science process

To understand the Azure Machine Learning service, let's consider how it fits into the machine learning development process illustrated below.



Environment setup

You start by creating a *workspace*, which is a place in Azure for you to store machine learning work. You can create a workspace in the Azure portal or from within Python code. An experiment object is created within the workspace to store information about runs for the models that you train and test. You can have multiple experiment objects in a workspace.

The Azure Machine Learning service allows you to interact with a workspace by using your preferred integrated development environment (IDE), such as a local Jupyter notebook, PyCharm, or a notebook in Azure Notebooks (a cloud version of Jupyter Notebook). As you will see later in this module, it's easy to configure the environment.

Data preparation

Before you can train a model, you must explore and analyze the source data to determine its quality and to select data for model features. Typically, this involves statistical analysis and the use of visualizations. Then, in the data wrangling step, you clean up the data and apply transformations to prepare it for use in model training.

You can use whatever Python modules you like for data preparation, including Pandas or the Azure Machine Learning Data Preparation SDK called **Azureml.dataprep**.

Experimentation

Experimentation is the iterative process of model training and testing. Open-source packages like Scikit-learn, TensorFlow, and others are supported.

After building a model, you can train it locally or on a remote computer.

A key feature of the Azure Machine Learning service is the ability to run model training and evaluation in Azure containers. It's simple to monitor remote model execution and retrieve output by using the Azureml package. You also must create and configure a compute target object, which is used to provision computing resource.

After you have the model you want to use in production, you register the model in the workspace.

Deployment

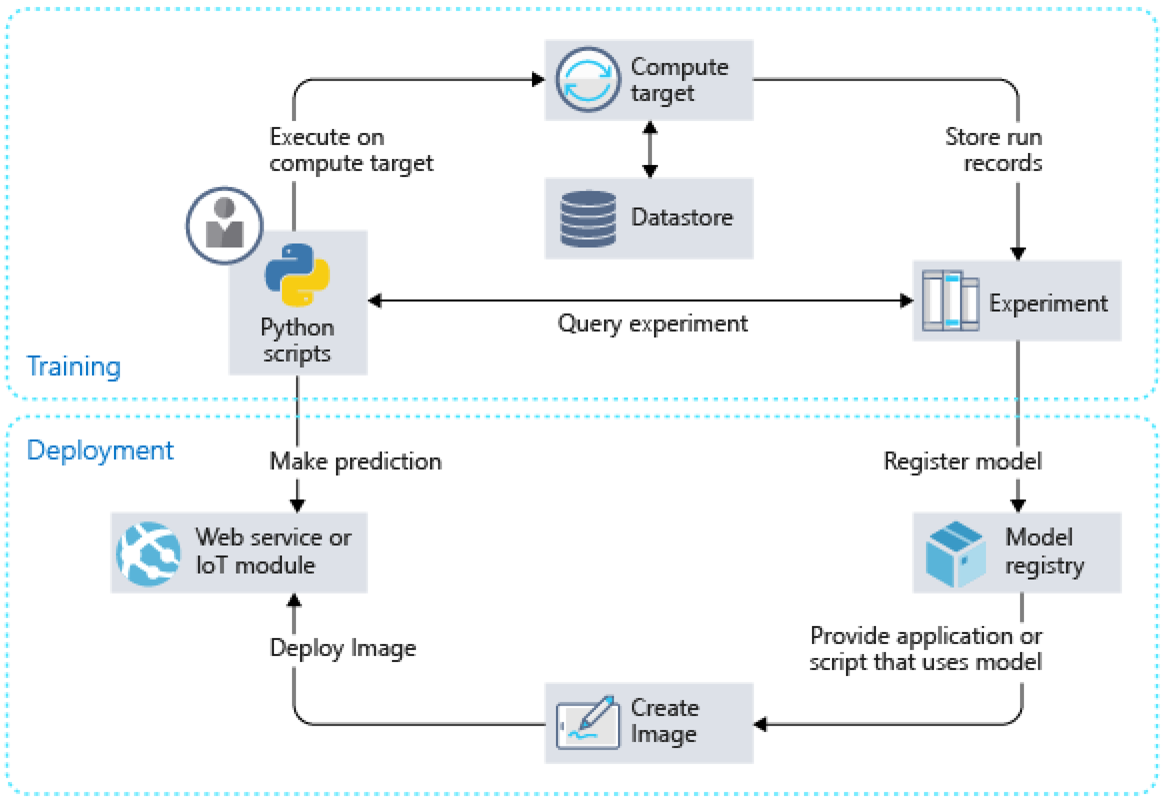
After ensuring that the model runs correctly in the local environment and is performing at the accuracy level that you want, you can deploy the model.

You will create a Docker image and then deploy it to Azure Container Instances. Note: Other target environments are available, including Azure Kubernetes Service (AKS), Azure IoT Edge, and a field programmable gate array (FPGA). For the deployment, you need the following files:

- The score scripts file is needed to decide how to run the model.
- The environment file is needed to specify package dependencies, which are important when using open-source packages.
- The configuration file is needed to request an appropriate amount of resources for the container.

Create a machine learning experiment

Let's consider the key components of the Azure Machine Learning service. The following diagram illustrates the data science process within the Azure Machine Learning service.



The process usually is under the umbrella of one workspace. You use Python to request compute targets and to query an experiment during the training phase. Once a model has been trained and registered, an image will be created for deployment. You can then use Python to run a deployed model in the Web service or IoT module.

Let's explore some of the key components from the above process.

What is a Workspace?

A *workspace* is the top-level resource for the Azure Machine Learning service. It serves as a hub for building and deploying models. You can create a workspace in the Azure portal, or you can create and access it by using Python in an IDE of your choice.

All models must be registered in the workspace for future use. Together with the scoring scripts, you create an image for deployment.

The workspace stores experiment objects that are required for each model you create. Additionally, it saves your compute targets. You can track training runs, and you can retrieve logs, metrics, output, and scripts with ease. This information is important for model evaluation and selection.

What is an Image?

As you might recall, an image has three key components:

1. A model and scoring script or application
2. An environment file that declares the dependencies that are needed by the model, scoring script, or application
3. A configuration file that describes the necessary resources to execute the model

What is a Datastore?

A *datastore* is an abstraction over an Azure Storage account. Each workspace has a registered, default datastore that you can use right away, but you can register other Azure Blob or File storage containers as a datastore.

What is a Pipeline?

A machine learning *pipeline* is a tool to create and manage workflows during a data science process, which typically includes data manipulation, model training and testing, and deployment phases. Each step of the process can run unattended in different compute targets, which makes it easier to allocate resources.

What is a Compute target?

A *compute target* is the compute resource to run a training script or to host service deployment. It's attached to a workspace. Other than the local machine, users of the workspace share compute targets.

What is a deployed Web service?

For a deployed *Web service*, you have the choices of Container Instances, AKS, or FPGAs. With your model, script, and associated files all set in the image, you can create a Web service.

What is an IoT module?

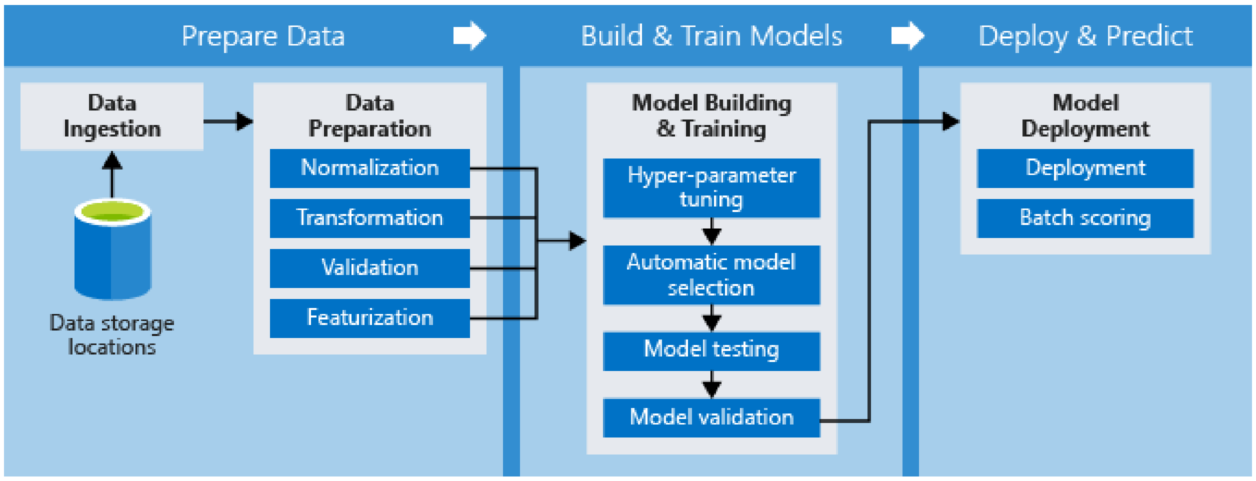
The *IoT module* is a Docker container. As with a Web service, you will need your model, associated script or application, and any additional dependencies. See IoT Edge (https://docs.microsoft.com/azure/iot-edge/) to learn more about the service. It enables you to monitor a hosting device.

Creating a pipeline

The Python SDK provides interfaces to work with Azure Machine Learning pipelines. The SDK includes imperative constructs for the sequencing and parallelization of steps. The goal is to achieve optimized execution.

All the data sources, inputs, and outputs are strictly named so that they can be reused across pipelines. The recorded intermediate tasks and data accelerate team collaboration and communication.

The following diagram is an example of a pipeline:



Here are few features of Azure Machine Learning pipelines:

- You can schedule tasks and execution, which frees up data scientists' time, especially during the data preparation phase.
- You have the flexibility to allocate compute targets for individual steps and to coordinate multiple pipelines.
- You can reuse pipeline scripts, and you can customize them for different processes; for example, in retraining and batch scoring.
- You can record and manage all input, output, intermediate tasks, and data.