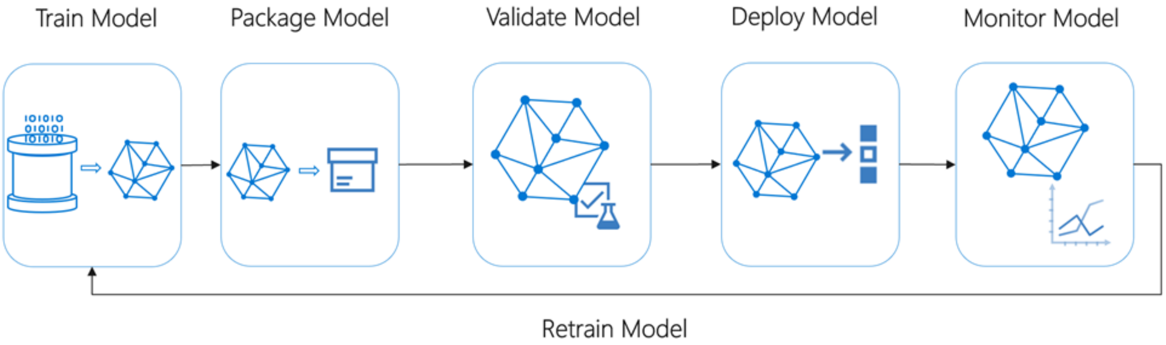
How to deploy machine learning models with Azure Machine Learning

In this article you will learn to deploy your machine learning models with Azure Machine Learning. Model deployment is the method by which you integrate a machine learning model into an existing production environment in order to start using it to make practical business decisions based on data.

Azure Machine Learning service is a cloud service that you use to train, deploy, automate, and manage machine learning models, all at the broad scale that the cloud provides. The service fully supports open-source technologies such as PyTorch, TensorFlow, and scikit-learn and can be used for any kind of machine learning, from classical ml to deep learning, supervised and unsupervised learning.

Moreover, Azure Machine Learning service introduces a new capability to help simplify model deployment process used in your machine learning lifecycle:



Some data scientists have difficulty getting an ML model prepared to run in a production system. To alleviate this, Azure Machine Learning can help you package and debug your machine learning models locally, prior to pushing them to the cloud. This should greatly reduce the inner loop time required to iterate and arrive at a satisfactory inferencing service, prior to the packaged model reaching the datacenter.

The deployment workflow is similar regardless of where you deploy your model:

1. Register the model.
2. Prepare to deploy (specify assets, usage, compute target)
3. Deploy the model to the compute target.
4. Consume the deployed model, also called web service.

1. Register your model

Register your machine learning models in your Azure Machine Learning workspace. The model can come from Azure Machine Learning or can come from somewhere else. The following examples demonstrate how to register a model from file:

Register a model from an Experiment Run

- Scikit-learn example using the SDK

```
model = run.register_model(model_name='sklearn_mnist', model_path='outputs/skl
print(model.name, model.id, model.version, sep='\t')
```

- Using the CLI

```
az ml model register -n sklearn_mnist  --asset-path outputs/sklearn_mnist_mode
```

- Using VS Code

Register models using any model files or folders with the Visual Studio Code extension.

Register an externally created model

You can register an externally created model by providing a local path to the model. You can provide either a folder or a single file.

- ONNX example with the Python SDK

```
onnx_model_url = "https://www.cntk.ai/OnnxModels/mnist/opset_7/mnist.tar.gz"
urllib.request.urlretrieve(onnx_model_url, filename="mnist.tar.gz")
!tar xvzf mnist.tar.gz

model = Model.register(workspace = ws,
                       model_path ="mnist/model.onnx",
                       model_name = "onnx_mnist",
                       tags = {"onnx": "demo"},
                       description = "MNIST image classification CNN from ONNX
```

- Using the CLI

```
az ml model register -n onnx_mnist -p mnist/model.onnx
```

2. Prepare to deploy

To deploy as a web service, you must create an inference configuration (InferenceConfig) and a deployment configuration. The entry script receives data submitted to a deployed web service and passes it to the model. It then takes the response returned by the model and returns that to the client.

The script contains two functions that load and run the model:

- **init():** Typically this function loads the model into a global object. This function is run only once when the Docker container for your web service is started.
- **run(input_data):** This function uses the model to predict a value based on the input data. Inputs and outputs to the run typically use JSON for serialization and de-serialization. You can also work with raw binary data. You can transform the data before sending to the model, or before returning to the client.

3. Deploy to target

The following table provides an example of creating a deployment configuration for each compute target:

Compute target	Deployment configuration example
Local	deployment_config = LocalWebservice.deploy_configuration(port=8890)
Azure Container Instance	deployment_config = AciWebservice.deploy_configuration(cpu_cores = 1, memory_gb = 1)
Azure Kubernetes Service	deployment_config = AksWebservice.deploy_configuration(cpu_cores = 1, memory_gb = 1)

Let’s see together the example of using an existing AKS cluster using the Azure Machine Learning SDK, CLI, or the Azure portal. If you already have an AKS cluster attached, you can deploy to it:

- **Using the SDK**

```
aks_target = AksCompute(ws,"myaks")
```

> **Note:** If deploying to a cluster configured for dev/test, ensure that it was created with enough cores and memory to handle this deployment configuration. Remember that memory is also used by things such as dependencies and AML components.

```
deployment_config = AksWebservice.deploy_configuration(cpu_cores = 1, memory_gb

service = Model.deploy(ws, "aksservice", [model], inference_config, deployment_

service.wait_for_deployment(show_output = True)

print(service.state)
print(service.get_logs())
```

- **Using the CLI**

```
az ml model deploy -ct myaks -m mymodel:1 -n aksservice -ic inferenceconfig.jso
```

- **Using VS Code:** You can also deploy to AKS via the VS Code extension, but you'll need to configure AKS clusters in advance.

4. Consume web services

Every deployed web service provides a REST API, so you can create client applications in a variety of programming languages. If you have enabled authentication for your service, you need to provide a service key as a token in your request header.

Here is an example of how to invoke your service in Python:

```
import requests
import json
```

```

headers = {'Content-Type':'application/json'}

if service.auth_enabled:
    headers['Authorization'] = 'Bearer '+service.get_keys()[0]

print(headers)

test_sample = json.dumps({'data': [
    1,2,3,4,5,6,7,8,9,10],
    10,9,8,7,6,5,4,3,2,1]
  ]
 }
)

response = requests.post(service.scoring_uri, data=test_sample, headers=header
print(response.status_code)
print(response.elapsed)
print(response.json())
```

You can send data to this API and receive the prediction returned by the model. The general workflow for creating a client that uses a machine learning web service is:

1. Use the SDK to get the connection information.
2. Determine the type of request data used by the model.
3. Create an application that calls the web service.

Conclusion

In this article you learnt the first steps of how to deploy your machine learning models with Azure Machine Learning. Azure Machine Learning can be used intensively across various notebooks for tasks relating to AI model development, such as:

- Hyperparameter tuning

- Tracking and monitoring metrics to enhance the model creation process
- Scaling up and out on compute like DSVM and Azure ML Compute
- Submitting pipelines

Learn more at:

- Azure Machine Learning Service (https://docs.microsoft.com/en-us/azure/machine-learning/service/?WT.mc_id=educative-article-lazzeri)
- Azure Machine Learning for Visual Studio Code (https://marketplace.visualstudio.com/items?itemName=ms-toolsai.vscode-ai&WT.mc_id=educative-article-lazzeri)
- Get Started with Azure Machine Learning (https://azure.microsoft.com/en-us/trial/get-started-machine-learning/?WT.mc_id=educative-article-lazzeri)
- Deploy Models with Azure Machine Learning (https://docs.microsoft.com/azure/machine-learning/service/how-to-deploy-and-where?WT.mc_id=educative-article-lazzeri)
- Azure Machine Learning Notebooks (https://github.com/Azure/MachineLearningNotebooks?WT.mc_id=educative-article-lazzeri)

About: Francesca Lazzeri

Francesca Lazzeri, PhD is Senior Machine Learning Scientist at Microsoft on the Cloud Advocacy team and expert in big data technology innovations and the applications of machine learning-based solutions to real-world problems. Her research has spanned the areas of machine learning, statistical modeling, time series econometrics and forecasting, and a range of industries – energy, oil and gas, retail, aerospace, healthcare, and professional services.

Before joining Microsoft, she was Research Fellow in Business Economics at Harvard Business School, where she performed statistical and econometric analysis within the Technology and Operations Management Unit. At Harvard, she worked on multiple patent, publication and social network data-driven projects to investigate and measure the impact of external knowledge networks on companies' competitiveness and innovation.

Francesca periodically teaches applied analytics and machine learning classes at universities and research institutions around the world. She is Data Science mentor for PhD and Postdoc students at the Massachusetts Institute of Technology, and speaker at academic and industry conferences - where she shares her knowledge and passion for AI, machine learning, and coding.

Twitter: @frlazzeri - https://twitter.com/frlazzeri (https://twitter.com/frlazzeri)

LinkedIn: https://www.linkedin.com/in/francescalazzeri/ (https://www.linkedin.com/in/francescalazzeri/)

Medium: https://medium.com/@francescalazzeri (https://medium.com/@francescalazzeri)

Further Readings

Course Track: Become a Machine Learning Engineer (https://www.educative.io/m/become-a-machine-learning-engineer)

Course: Grokking Data Science (https://www.educative.io/courses/grokking-data-science)

Article: Machine learning 101 & data science: Tips from an industry expert (https://www.educative.io/blog/machine-learning-for-data-science)

Article: How to ace your next ML interview (https://www.educative.io/blog/ml-interview)

Article: How and why to become a machine learning engineer (https://www.educative.io/blog/how-and-why-to-become-a-machine-learning-engineer)

Article: The practical approach to machine learning for software engineers (https://www.educative.io/blog/the-practical-approach-to-machine-learning-for-software-engineers)

Article: The disconnect b/w industry deep learning and university courses (https://www.educative.io/blog/ml-industry-university)

Article: My experience working with ML at Google and Microsoft (https://www.educative.io/blog/ml-microsoft-and-google)

LEARN

Courses (/explore)

Edpresso (/edpresso)

Blog (/blog)

For Students (/github-students)

Subscriptions (/subscription)

CodingInterview.com (//codinginterview.com/)

CONTRIBUTE

Become An Author (/authors)

Published Authors (/published-authors)

Become An Affiliate (/affiliate)

LEGAL

Privacy Policy (/privacy)

Terms of Service (/terms)

Enterprise Terms of Service (/enterprise-terms)

MORE

Team (/team)

Careers (//angel.co/educativeinc/jobs)

Business (/business)

FAQ (/courses/educative-faq/)

Contact Us (/contactUs)

SOCIAL

(//facebook.com/educativeinc) (//linkedin.com/company/educative-inc/) (//twitter.com/educativeinc)