

Advanced Machine Learning on Azure

Francesca Lazzeri, PhD

 @frlazzeri

 aka.ms/AzureMLGithub

 medium.com/@francescalazzeri

Agenda

2 components of model operationalization



Model deployment
workflow



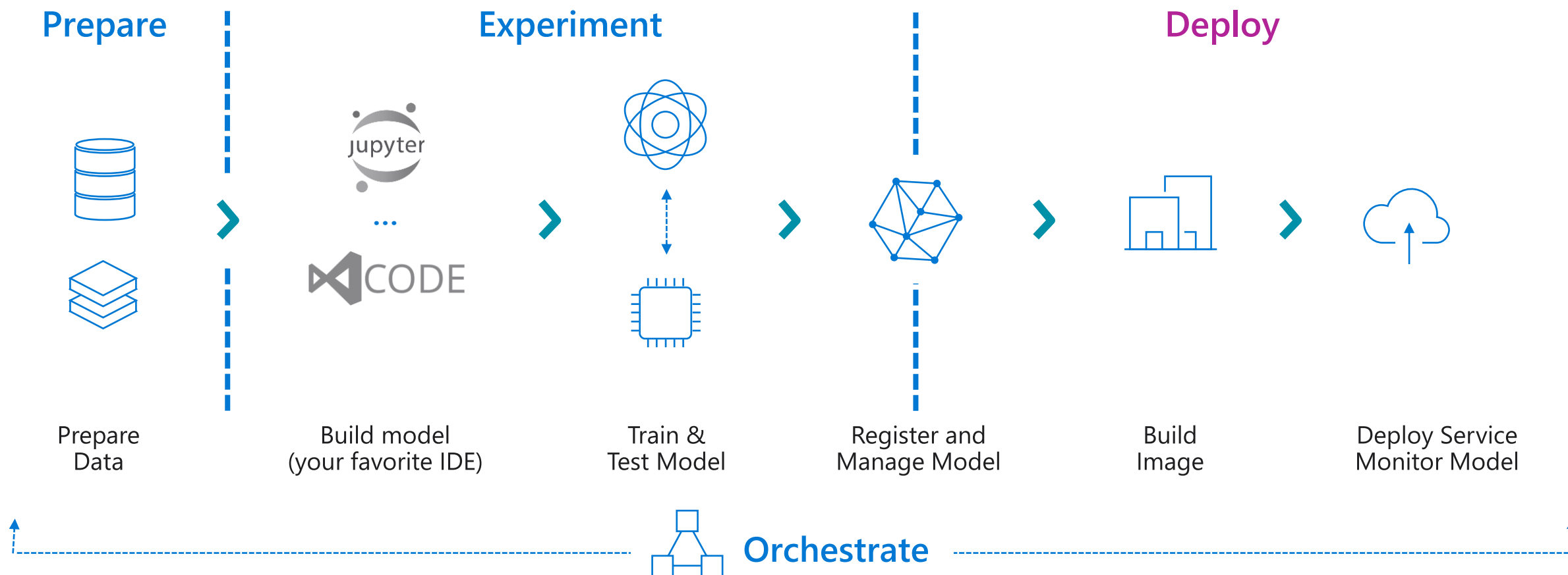
Automated machine
learning



Model deployment workflow

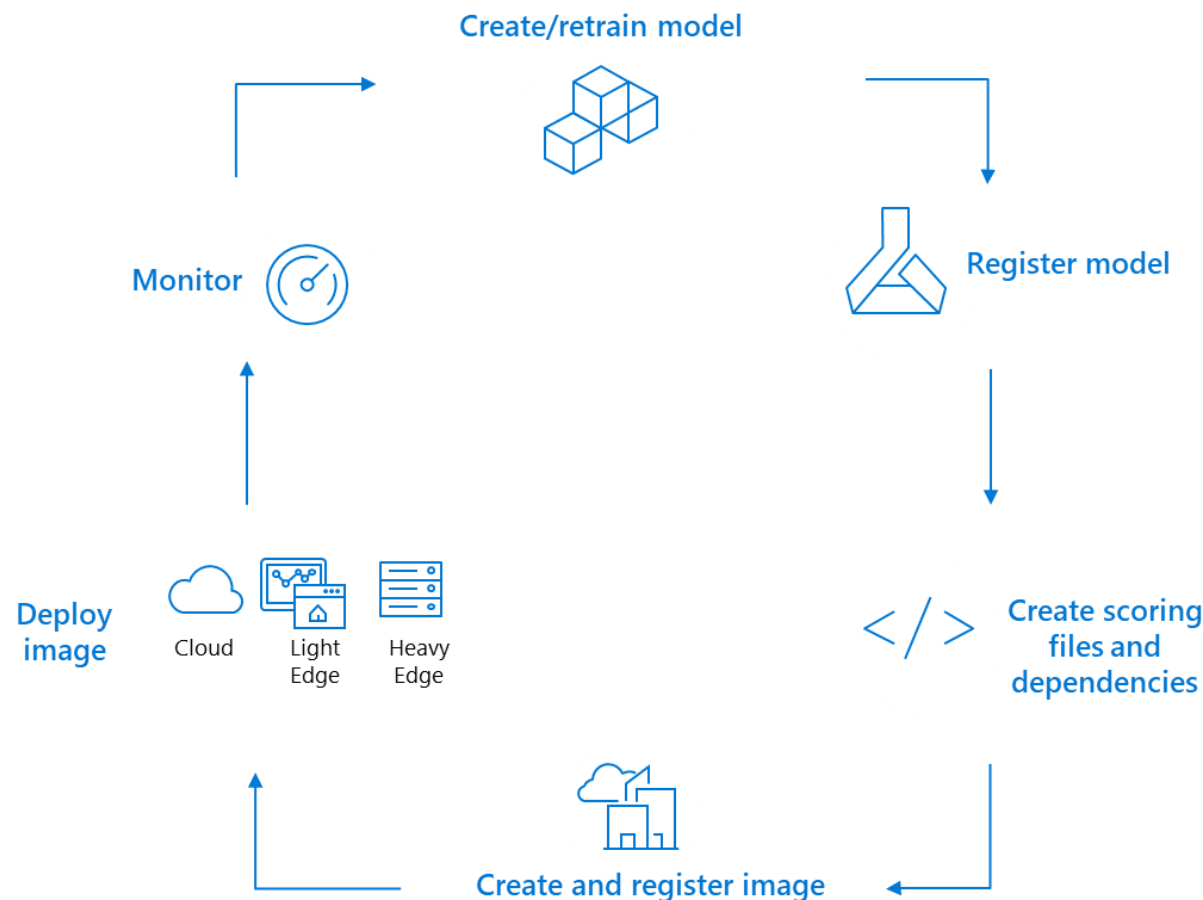
Machine Learning

Typical End To End Process



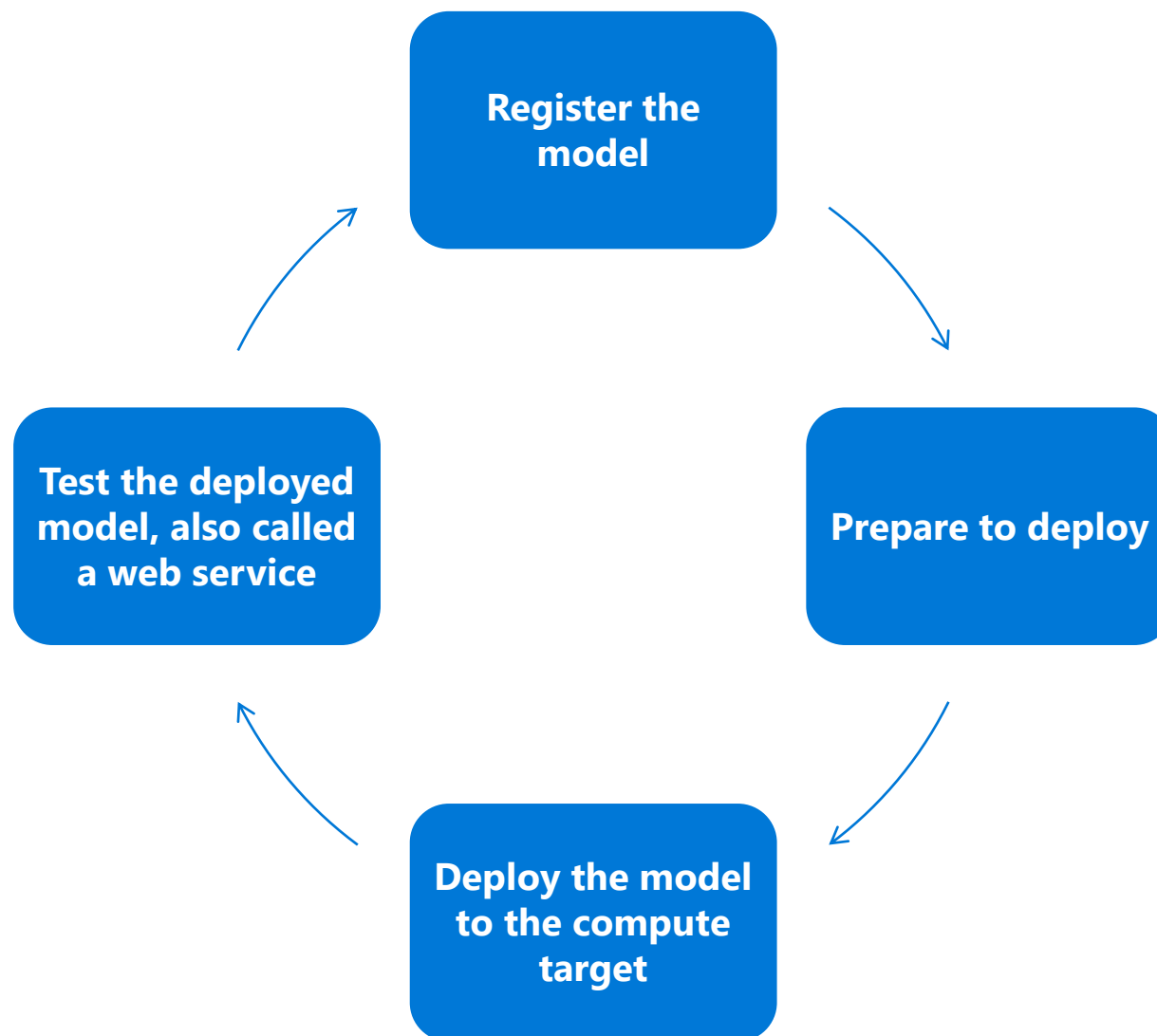
Manage model lifecycle

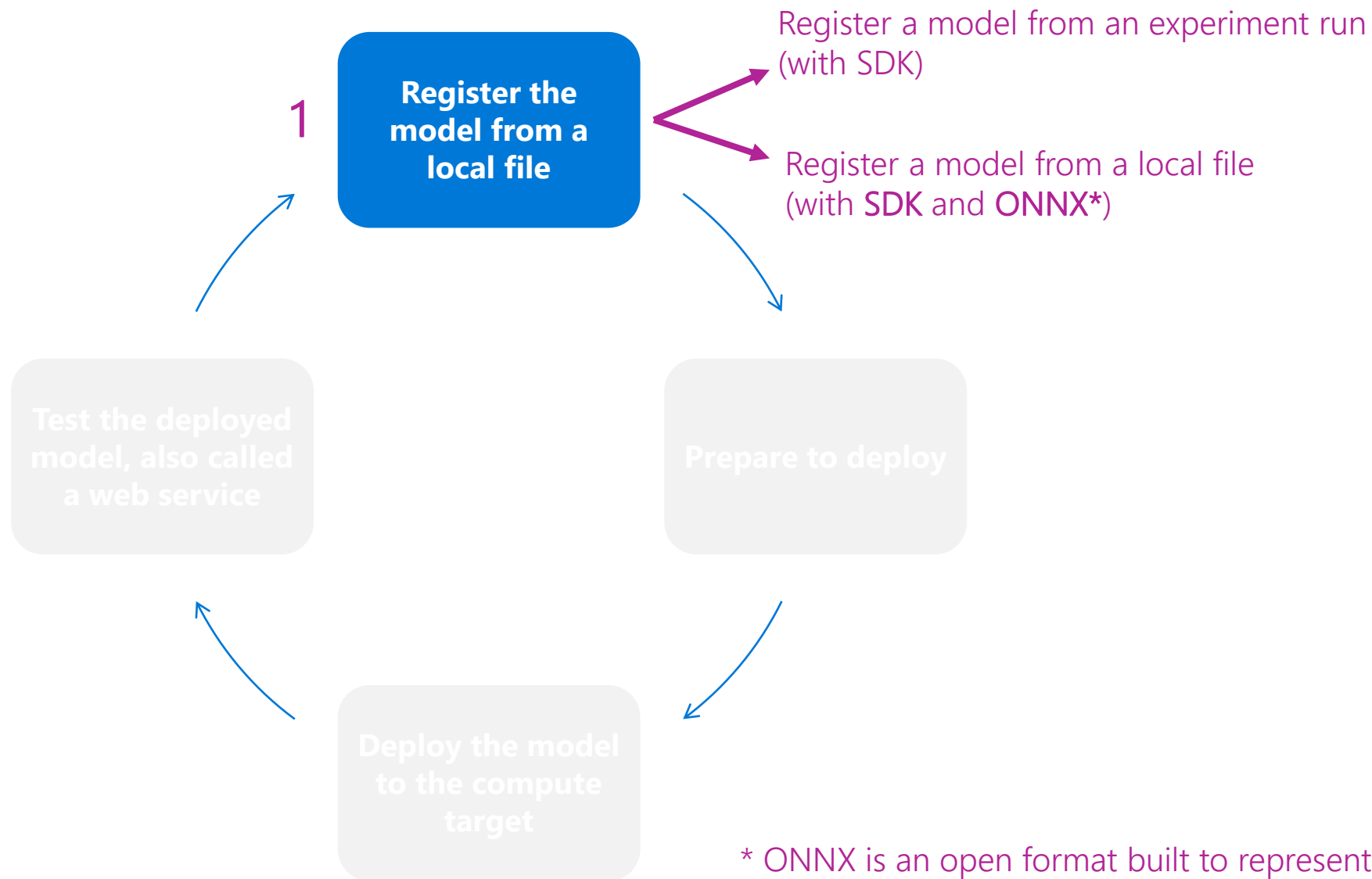
- **Track model versions and metadata** with a centralized model registry
- **Leverage containers** to capture runtime dependencies for inference
- Leverage an **orchestrator** like Kubernetes to provide scalable inference
- Capture **model telemetry** – health, performance, inputs/outputs
- Encapsulate each step in the lifecycle to enable **CI/CD and DevOps**
- Automatically **optimize models** to take advantage of hardware acceleration



Model deployment workflow

The workflow is similar no matter [where you deploy](#) your model:





* ONNX is an open format built to represent machine learning models - <https://onnx.ai/>

Step 1 – Register the model (Run Object)

```
model = run.register_model(model_name='sklearn_mnist',  
                           tags={'area': 'mnist'},  
                           model_path='outputs/sklearn_mnist_model.pkl')  
  
print(model.name, model.id, model.version, sep='\t')
```

OR...

Step 1 – Register the model (AutoMLRun Object)

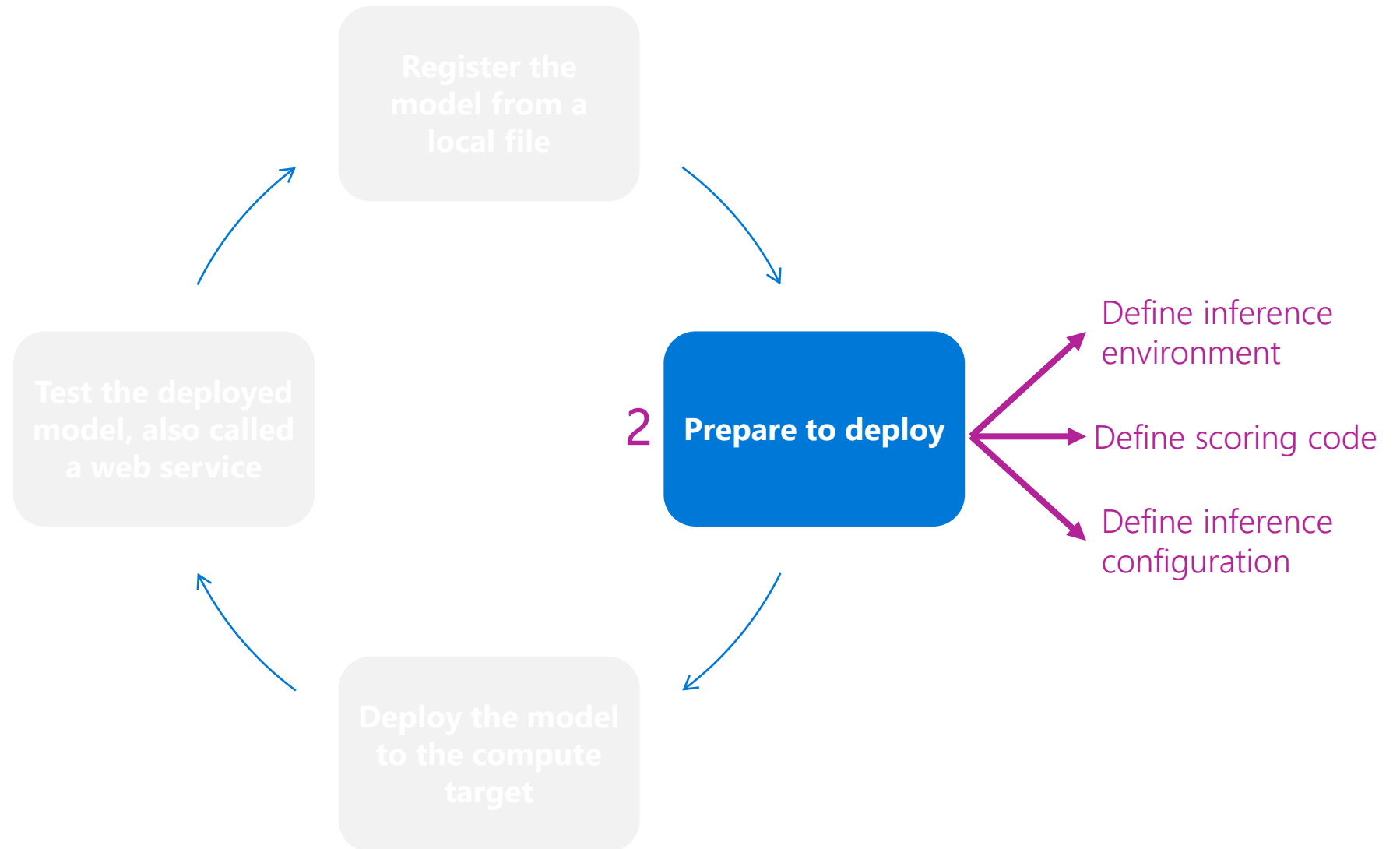
```
description = 'My AutoML Model'  
model = run.register_model(description = description,  
                           tags={'area': 'mnist'})  
  
print(run.model_id)
```


Step 1 – Register a model from a local file

```
import os
import urllib.request
from azureml.core.model import Model

# Download model
onnx_model_url = "https://www.cntk.ai/OnnxModels/mnist/opset_7/mnist.tar.gz"
urllib.request.urlretrieve(onnx_model_url, filename="mnist.tar.gz")
os.system('tar xvzf mnist.tar.gz')

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



Step 2.1 – Prepare to Deploy: define inference environment

```
name: project_environment
dependencies:
  - python=3.6.2
  - scikit-learn=0.20.0
  - pip:
    # You must list azureml-defaults as a pip dependency
    - azureml-defaults>=1.0.45
    - inference-schema[numpy-support]
```

YAML file



```
from azureml.core.environment import Environment
myenv = Environment.from_conda_specification(name = 'myenv',
                                           file_path = 'path-to-conda-specification-file')

myenv.register(workspace=ws)
```

Step 2.2 – Prepare to Deploy: define scoring code

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 of the run typically use JSON for serialization and deserialization. You can also work with raw binary data. You can transform the data before sending it to the model or before returning it to the client.

Load model files in your entry script

AZUREML_MODEL_DIR: An environment variable containing the path to the model location.

Model.get_model_path: An API that returns the path to model file using the registered model name.

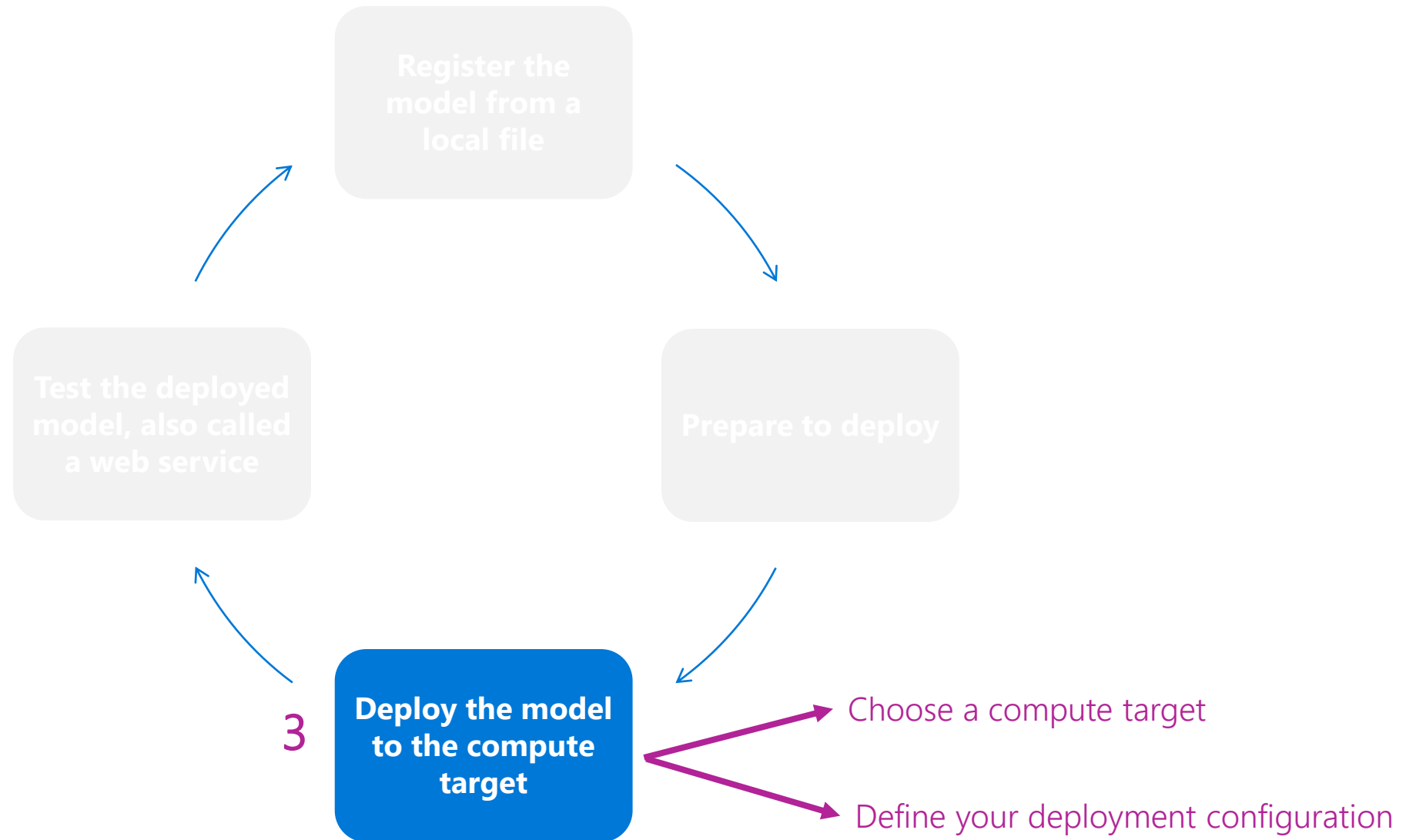
Step 2.3 – Prepare to Deploy: define inference environment

```
from azureml.core.environment import Environment
from azureml.core.model import InferenceConfig

myenv = Environment.get(workspace=ws, name='myenv', version='1')
inference_config = InferenceConfig(entry_script='path-to-score.py',
                                   environment=myenv)
```

InferenceConfig class

- Represents configuration settings for a custom environment used for deployment.
- Inference configuration is an input parameter for [Model](#) deployment-related actions:
 - [deploy\(workspace, name, models, inference_config=None, deployment_config=None, deployment_target=None, overwrite=False\)](#)
 - [profile\(workspace, profile_name, models, inference_config, input_data=None, input_dataset=None, cpu=None, memory_in_gb=None, description=None\)](#)
 - [package\(workspace, models, inference_config=None, generate_dockerfile=False\)](#)



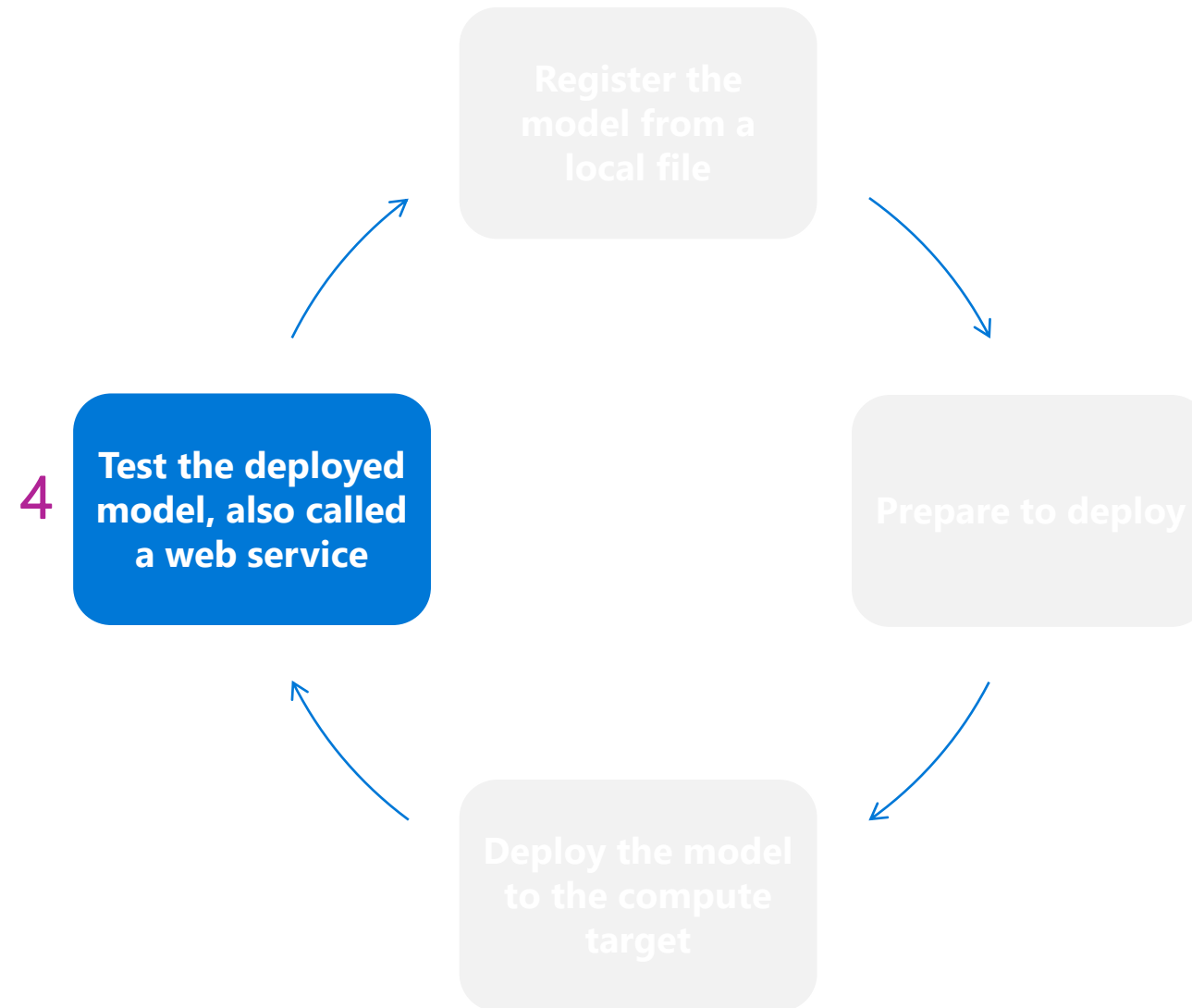
Step 3.1 – Deploy the model to the compute target:
choose a compute target

| Compute target | Used for | GPU support | FPGA support | Description |
|-----------------------------------------------------|-------------------------------|---------------------------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Local web service | Testing/debugging | | | Use for limited testing and troubleshooting. Hardware acceleration depends on use of libraries in the local system. |
| Azure Machine Learning compute instance web service | Testing/debugging | | | Use for limited testing and troubleshooting. |
| Azure Kubernetes Service (AKS) | Real-time inference | Yes (web service deployment) | Yes | Use for high-scale production deployments. Provides fast response time and autoscaling of the deployed service. Cluster autoscaling isn't supported through the Azure Machine Learning SDK. To change the nodes in the AKS cluster, use the UI for your AKS cluster in the Azure portal. AKS is the only option available for the designer. |
| Azure Container Instances | Testing or development | | | Use for low-scale CPU-based workloads that require less than 48 GB of RAM. |
| Azure Machine Learning compute clusters | (Preview) Batch inference | Yes (machine learning pipeline) | | Run batch scoring on serverless compute. Supports normal and low-priority VMs. |
| Azure Functions | (Preview) Real-time inference | | | |
| Azure IoT Edge | (Preview) IoT module | | | Deploy and serve ML models on IoT devices. |
| Azure Data Box Edge | Via IoT Edge | | Yes | Deploy and serve ML models on IoT devices. |

Step 3.2 – Deploy the model to the compute target: define your deployment configuration

| Compute target | Deployment configuration example |
|---------------------------|---------------------------------------------------------------------------------------------------|
| Local | <code>deployment_config = LocalWebservice.deploy_configuration(port=8890)</code> |
| Azure Container Instances | <code>deployment_config = AciWebservice.deploy_configuration(cpu_cores = 1, memory_gb = 1)</code> |
| Azure Kubernetes Service | <code>deployment_config = AksWebservice.deploy_configuration(cpu_cores = 1, memory_gb = 1)</code> |

```
from azureml.core.webservice import AciWebservice, AksWebservice, LocalWebservice
```

Step 4 – Test the deployed model, also called a web service - Request-response consumption

```
import requests
import json

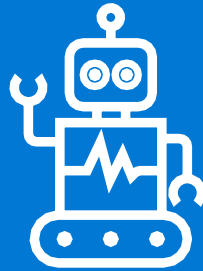
headers = {'Content-Type': 'application/json'}

if service.auth_enabled:
    headers['Authorization'] = 'Bearer '+service.get_keys()[0]
elif service.token_auth_enabled:
    headers['Authorization'] = 'Bearer '+service.get_token()[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=headers)
print(response.status_code)
print(response.elapsed)
print(response.json())
```

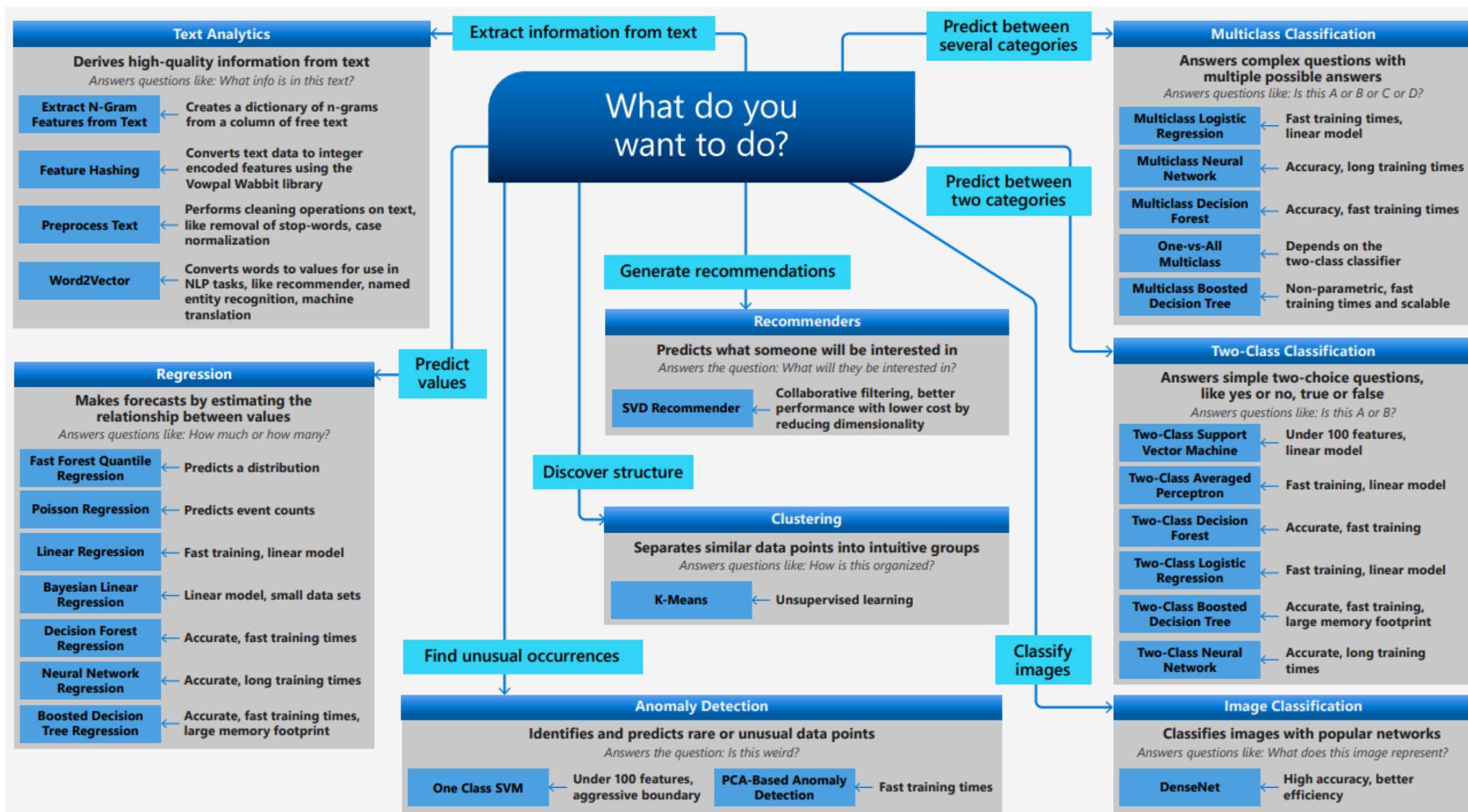


Automated Machine Learning
'simplifies' the creation and selection
of the optimal model

Complexity of Machine Learning

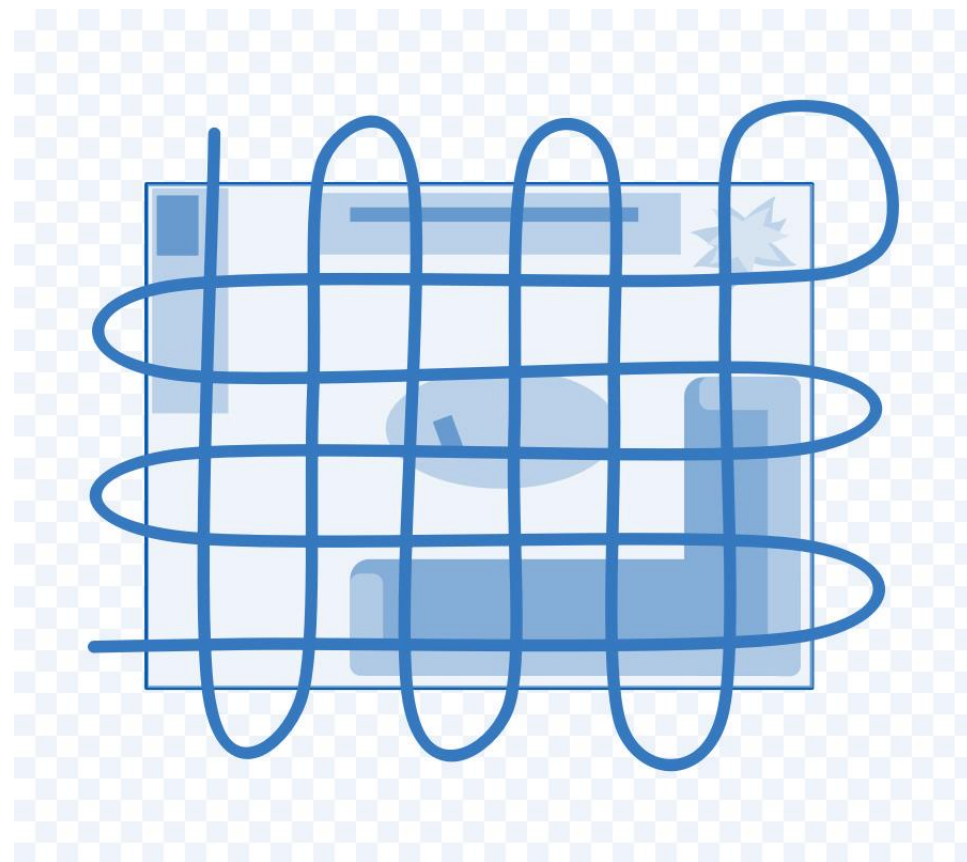
www.aka.ms/algorithmcheatsheet

 @frlazzeri



Challenges with Hyperparameter Selection

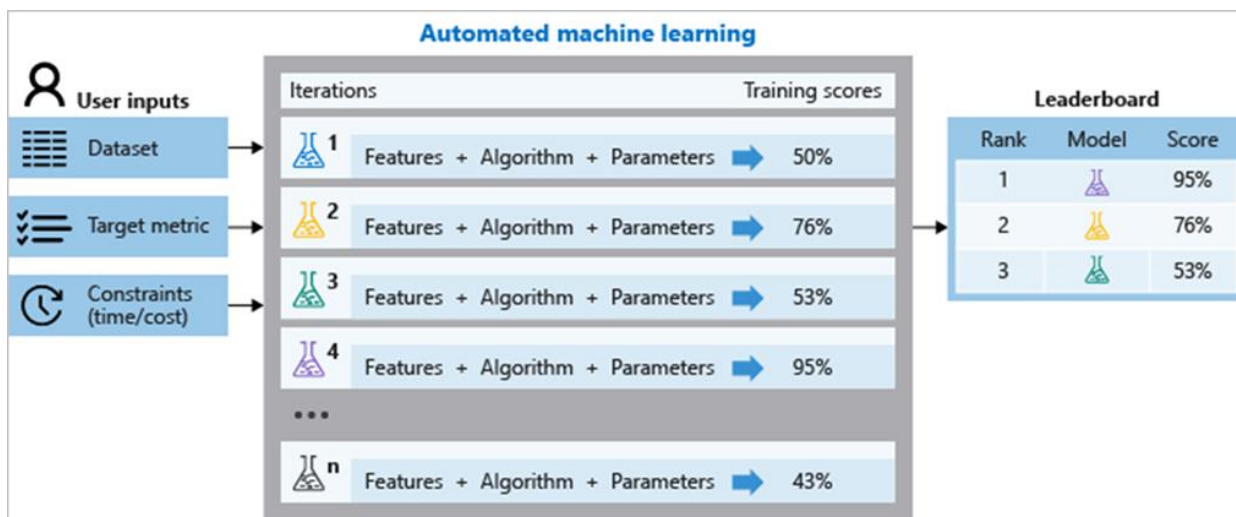
- The search space to explore—i.e. evaluating all possible combinations—is huge.
- Sparsity of good configurations.
Very few of all possible configurations are optimal.
- Evaluating each configuration is resource and time consuming.
- Time and resources are limited.



Automated Machine Learning

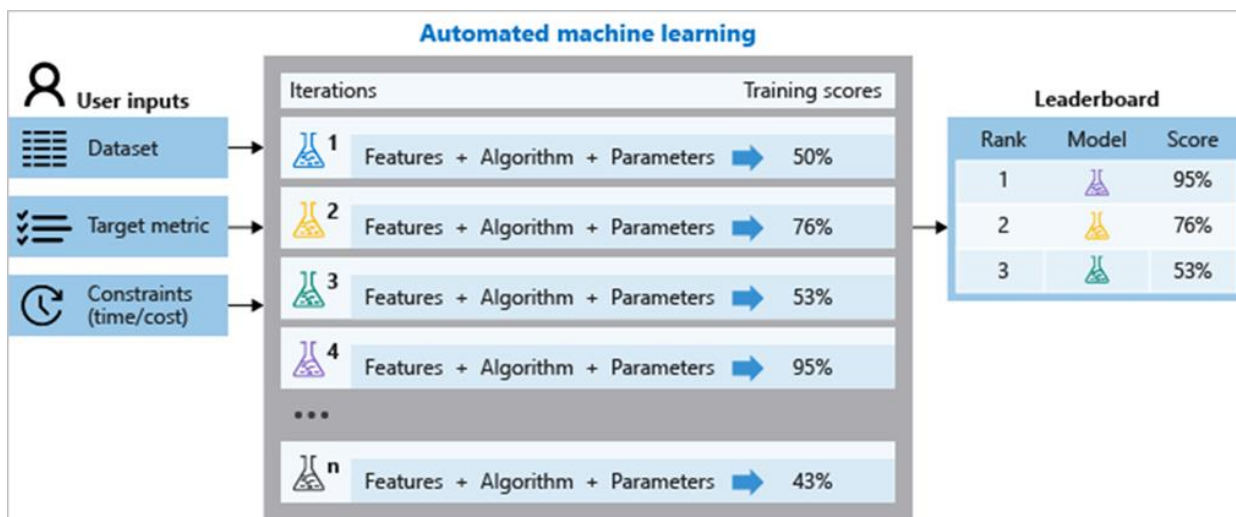
- Automated machine learning is the process of automating the time consuming, iterative tasks of **machine learning model development**.
- Data scientists, analysts and developers across industries can use automated ML to:
 - Implement machine learning solutions without extensive programming knowledge
 - Save time and resources
 - Leverage data science best practices
 - Provide agile problem-solving

How Automated ML works



- 1) Identify the ML problem to be solved: **classification, forecasting, or regression**
- 2) Specify the source and format of the labeled training data: **Numpy arrays** or **Pandas dataframe**
- 3) Configure the **compute target** for model training, such as your local computer, Azure Machine Learning Computes, remote VMs, or Azure Databricks
- 4) Configure the automated machine learning parameters that determine how many **iterations** over different models, **hyperparameter** settings, advanced **preprocessing/featurization**, and what **metrics** to look at when determining the best model
- 5) Submit the **training run**

How Automated ML works



- During training, the Azure Machine Learning service creates a number of in **parallel pipelines** that try different algorithms and parameters. It will stop once it hits the **exit criteria** defined in the experiment
- You can also inspect the **logged run information**, which contains metrics gathered during the run
- The training run produces a Python serialized object (**.pkl file**) that contains the model and data preprocessing
- While model building is automated, **you can also learn how important or relevant features are to the generated models**

Automated and Interpretable ML 



Demo



Useful Resources:

- ✓ Azure MLOps: aka.ms/AMLOps
- ✓ AzureML GitHub: aka.ms/AzureMLGitHub
- ✓ Azure Machine Learning service: aka.ms/AzureMLservice
- ✓ Get started with Azure ML: aka.ms/GetStartedAzureML
- ✓ Automated Machine Learning Documentation: aka.ms/AutomatedMLDocs
- ✓ What is Automated Machine Learning? aka.ms/AutomatedML
- ✓ Model Interpretability with Azure ML Service: aka.ms/AzureMLModelInterpretability
- ✓ Algorithm Cheat Sheet: aka.ms/AlgorithmCheatSheet
- ✓ How to select Machine Learning algorithms: aka.ms/SelectAlgos
- ✓ Deep Learning vs Machine Learning: aka.ms/DeepLearningVSMachineLearning

Thank you!

Francesca Lazzeri, PhD



@frlazzeri



aka.ms/AzureMLGithub



medium.com/@francescalazzeri

