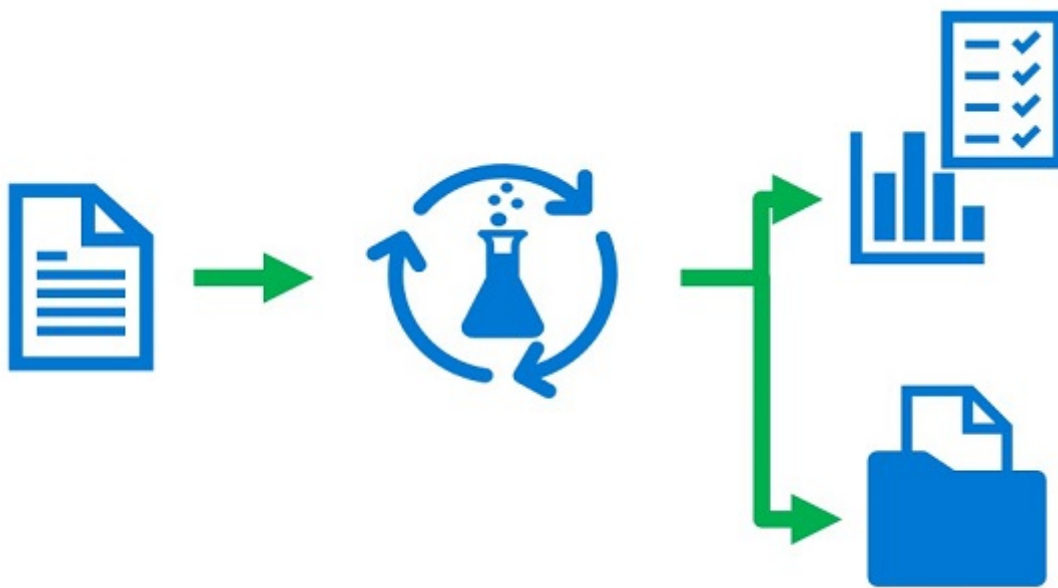


# Azure Machine Learning experiments

5 minutes

Like any scientific discipline, data science involves running *experiments*; typically to explore data or to build and evaluate predictive models. In Azure Machine Learning, an experiment is a named process, usually the running of a script or a pipeline, that can generate metrics and outputs and be tracked in the Azure Machine Learning workspace.



An experiment can be run multiple times, with different data, code, or settings; and Azure Machine Learning tracks each run, enabling you to view run history and compare results for each run.

## The Experiment Run Context

When you submit an experiment, you use its *run context* to initialize and end the experiment run that is tracked in Azure Machine Learning, as shown in the following code sample:

Python

Copy

```
from azureml.core import Experiment

# create an experiment variable
experiment = Experiment(workspace = ws, name = "my-experiment")

# start the experiment
run = experiment.start_logging()
```

```
# experiment code goes here

# end the experiment
run.complete()
```

After the experiment run has completed, you can view the details of the run in the **Experiments** tab in Azure Machine Learning studio.

## Logging Metrics and Creating Outputs

Experiments are most useful when they produce metrics and outputs that can be tracked across runs.

### Logging Metrics

Every experiment generates log files that include the messages that would be written to the terminal during interactive execution. This enables you to use simple `print` statements to write messages to the log. However, if you want to record named metrics for comparison across runs, you can do so by using the **Run** object; which provides a range of logging functions specifically for this purpose. These include:

- **log**: Record a single named value.
- **log\_list**: Record a named list of values.
- **log\_row**: Record a row with multiple columns.
- **log\_table**: Record a dictionary as a table.
- **log\_image**: Record an image file or a plot.

**More Information:** For more information about logging metrics during experiment runs, see [Monitor Azure ML experiment runs and metrics](#) in the Azure Machine Learning documentation.

For example, following code records the number of observations (records) in a CSV file:

Python

 Copy

```
from azureml.core import Experiment
import pandas as pd

# Create an Azure ML experiment in your workspace
experiment = Experiment(workspace = ws, name = 'my-experiment')

# Start logging data from the experiment
run = experiment.start_logging()
```

```
# load the dataset and count the rows
data = pd.read_csv('data.csv')
row_count = (len(data))

# Log the row count
run.log('observations', row_count)

# Complete the experiment
run.complete()
```

## Retrieving and Viewing Logged Metrics

You can view the metrics logged by an experiment run in Azure Machine Learning studio or by using the **RunDetails** widget in a notebook, as shown here:

Python

 Copy

```
from azureml.widgets import RunDetails

RunDetails(run).show()
```

You can also retrieve the metrics using the **Run** object's **get\_metrics** method, which returns a JSON representation of the metrics, as shown here:

Python

 Copy

```
import json

# Get logged metrics
metrics = run.get_metrics()
print(json.dumps(metrics, indent=2))
```

The previous code might produce output similar to this:

JSON

 Copy

```
{
  "observations": 15000
}
```

## Experiment Output Files

In addition to logging metrics, an experiment can generate output files. Often these are trained machine learning models, but you can save any sort of file and make it available as an


output of your experiment run. The output files of an experiment are saved in its **outputs** folder.

The technique you use to add files to the outputs of an experiment depend on how you're running the experiment. The examples shown so far control the experiment lifecycle inline in your code, and when taking this approach you can upload local files to the run's **outputs** folder by using the **Run** object's **upload\_file** method in your experiment code as shown here:


Python	 Copy
<pre>run.upload_file(name='outputs/sample.csv', path_or_stream='./sample.csv')</pre>	

When running an experiment in a remote compute context (which we'll discuss later in this course), any files written to the **outputs** folder in the compute context are automatically uploaded to the run's **outputs** folder when the run completes.

Whichever approach you use to run your experiment, you can retrieve a list of output files from the **Run** object like this:

Python	 Copy
<pre>import json  files = run.get_file_names() print(json.dumps(files, indent=2))</pre>	

The previous code would produce output similar to this:

JSON	 Copy
<pre>[   "outputs/sample.csv" ]</pre>	

## Running a Script as an Experiment

You can run an experiment inline using the **start\_logging** method of the **Experiment** object, but it's more common to encapsulate the experiment logic in a script and run the script as an experiment. The script can be run in any valid compute context, making this a more flexible solution for running experiments as scale.

An experiment script is just a Python code file that contains the code you want to run in the experiment. To access the experiment run context (which is needed to log metrics) the script

must import the `azureml.core.Run` class and call its `get_context` method. The script can then use the run context to log metrics, upload files, and complete the experiment, as shown in the following example:

Python

 Copy

```
from azureml.core import Run
import pandas as pd
import matplotlib.pyplot as plt
import os

# Get the experiment run context
run = Run.get_context()

# load the diabetes dataset
data = pd.read_csv('data.csv')

# Count the rows and log the result
row_count = (len(data))
run.log('observations', row_count)

# Save a sample of the data
os.makedirs('outputs', exist_ok=True)
data.sample(100).to_csv("outputs/sample.csv", index=False, header=True)

# Complete the run
run.complete()
```

To run a script as an experiment, you must define a *script configuration* that defines the script to be run and the Python environment in which to run it. This is implemented by using a `ScriptRunConfig` object.

For example, the following code could be used to run an experiment based on a script in the `experiment_files` folder (which must also contain any files used by the script, such as the `data.csv` file in previous script code example):

Python

 Copy

```
from azureml.core import Experiment, ScriptRunConfig

# Create a script config
script_config = ScriptRunConfig(source_directory=experiment_folder,
                                script='experiment.py')

# submit the experiment
experiment = Experiment(workspace = ws, name = 'my-experiment')
run = experiment.submit(config=script_config)
run.wait_for_completion(show_output=True)
```

### ⓘ Note

An implicitly created **RunConfiguration** object defines the Python environment for the experiment, including the packages available to the script. If your script depends on packages that are not included in the default environment, you must associate the **ScriptRunConfig** with an **Environment** object that makes use of a **CondaDependencies** object to specify the Python packages required. Runtime environments are discussed in more detail later in this course.

---

## Next unit: Exercise - Run experiments

Continue >