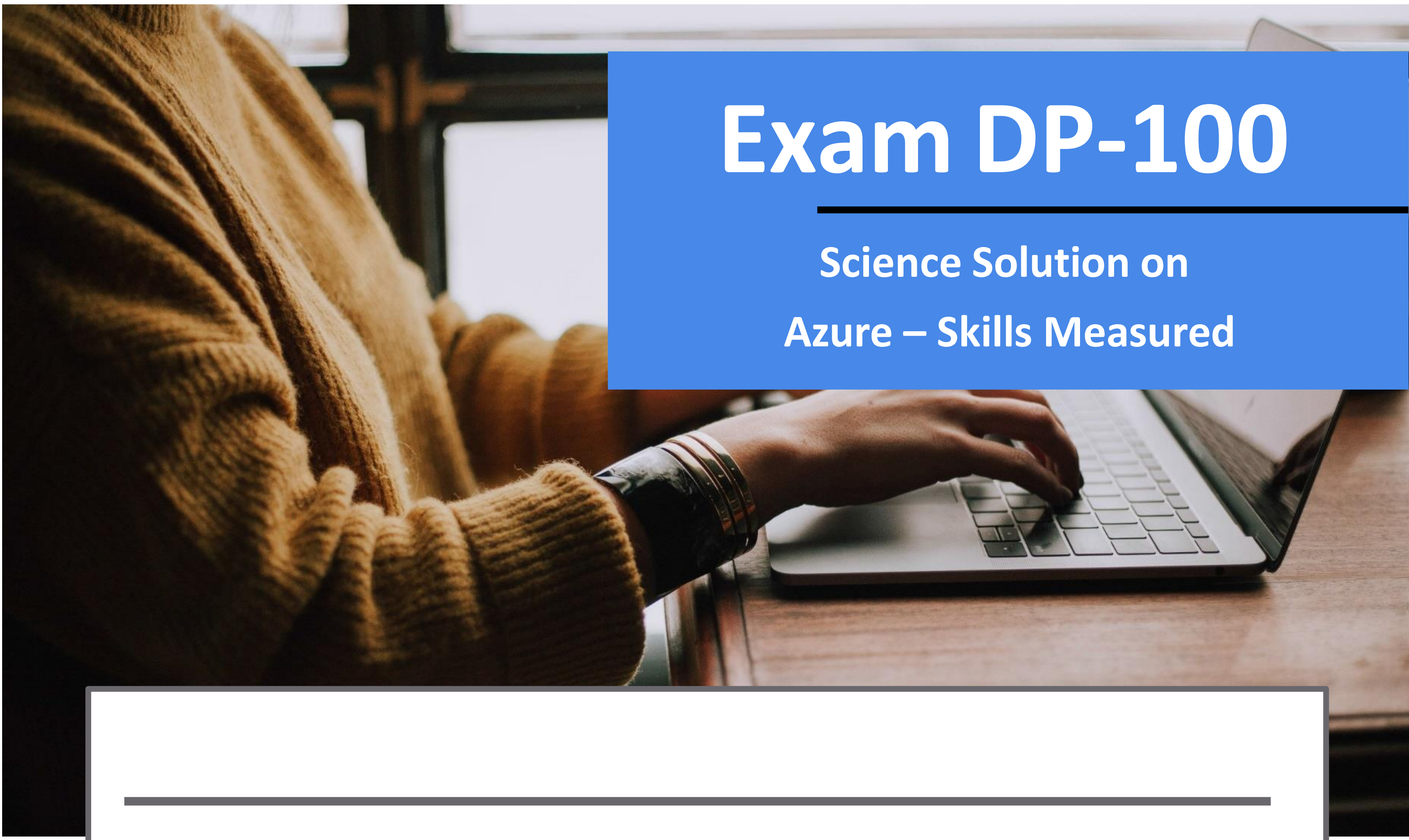
THE QUILL CONSULTANCY



Exam DP-100

Science Solution on Azure – Skills Measured

Address
Level 11, 39 Murray Street, Hobart,
Tasmania 7000 Australia

Phone
03 6234 3883

Email
quill@quill.com.au

Web
www.quill.com.au

Audience Profile

Candidates for the Azure Jo Associate certification should have subject matter expertise applying data science and machine learning to implement and run machine learning workloads on Azure

Responsibilities for this role include planning and creating a suitable working environment for data science workloads on Azure. You run data experiments and train predictive models. In addition, you manage, optimize, and deploy machine learning models into production.

A candidate for this certification should have knowledge and experience in data science and using Azure Machine Learning and Azure Databricks.

Contents

Contents

the Networking and Advanced sections to configure more settings for the workspace.

8. Review the settings and make any other changes or corrections. When you're satisfied with the settings, select Create.
9. To view the new workspace, select Go to resource.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-manage-workspace

Configure workspace settings

AZ security workspace-setting
Shows the workspace settings in your subscription - these settings let you control which work-space will hold your security data.

AZ security workspace-setting create
Creates a workspace settings in your subscription - these settings let you control which work-space will hold your security data.

AZ security workspace-setting delete
Deletes the workspace settings in your subscription - this will make the security events on the subscription be reported to the default workspace.

AZ security workspace-setting list
Shows the workspace settings in your subscription - these settings let you control which work-space will hold your security data.

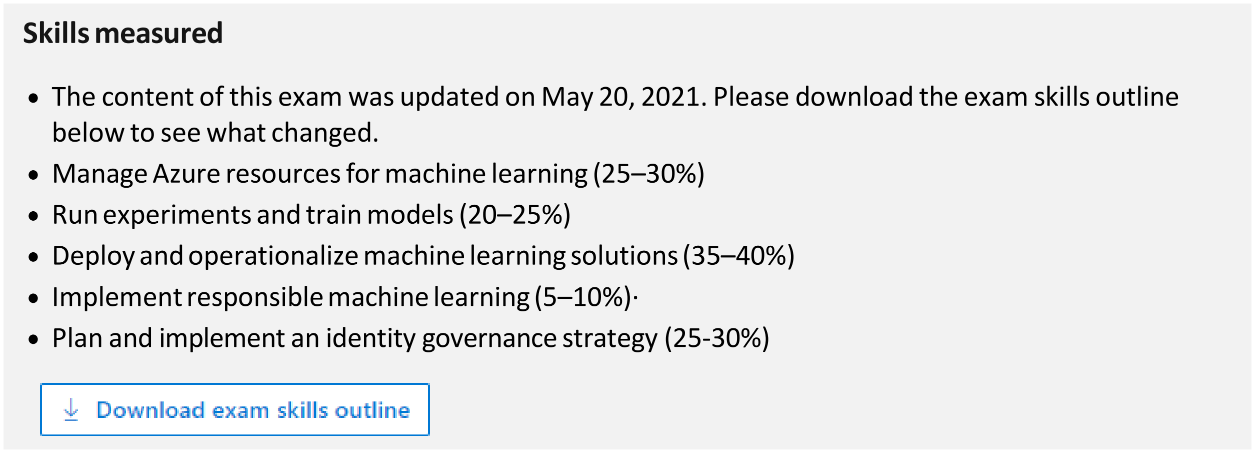
AZ security workspace-setting show
Shows the workspace settings in your subscription - these settings let you control which work-space will hold your security data.

https://docs.microsoft.com/en-us/cli/azure/security/workspace-setting?view=azure-cli-latest

How to use this guide

This guide is here to help you prepare and take the exam. It is designed to complement your existing learning and to help guide you in the areas of focus for the exam. You should use this as a framework to help fill in the blanks on information that you have.

We have developed the following content in direct alignment to the current Learning objectives. These can be viewed directly, by selecting the “Download exam skills outline” from the exam page at: https://docs.microsoft.com/en-us/learn/certifications/exams/dp-100



Skills measured

- The content of this exam was updated on May 20, 2021. Please download the exam skills outline below to see what changed.
- Manage Azure resources for machine learning (25–30%)
- Run experiments and train models (20–25%)
- Deploy and operationalize machine learning solutions (35–40%)
- Implement responsible machine learning (5–10%)·
- Plan and implement an identity governance strategy (25-30%)

There are loads of exciting and interesting topics we can begin to follow on from these core objectives, but remember for the exam we do need to stay focused and constrain ourselves to these key topics.

In the exam

The exam itself is quite straight forward with no complicated case studies or longwinded questions. The majority of the questions will be “Multiple Choice” or “Choose all that apply” type of questions. You may also come across some “Drag and Drop” questions where you need to place answers in order. The key thing to note is that all of the questions will have the answer in front of you.

Remembering that all of the answers are presented to you, you need to make sure that you answer each question. There is no loss of marks for incorrect answers, so even if you don't know the answer, you should attempt it.

The exam itself will have between 40 and 50 questions, depending on the pool of questions that have been allocated. You will have 60 minutes to complete the exam. As you can see you will need to move at a steady pace throughout. Don't get too stuck on any question, instead select your answer and then mark the question for "Review". Then if you have time at the end of the exam you can go back and review these questions.

Key Learning Objectives

Manage Azure resources for machine learning (25-30%)

Create an Azure Machine Learning workspace

Create an Azure Machine Learning workspace

Manage Azure Machine Learning workspaces in the portal or with the Python SDK

In this article, you create, view, and delete Azure Machine Learning workspaces for Azure Machine Learning, using the Azure portal or the SDK for Python

As your needs change or requirements for automation increase you can also manage workspaces using the CLI, or via the VS Code extension.

Portal

1. Sign in to the Azure portal by using the credentials for your Azure subscription.
2. In the upper-left corner of Azure portal, select + Create a resource.
3. Use the search bar to find Machine Learning.
4. Select Machine Learning.
5. In the Machine Learning pane, select Create to begin.
6. Provide the following information to configure your new workspace:
7. When you're finished configuring the workspace, select Review + Create. Optionally, use

Manage a workspace by using Azure Machine Learning studio

Limitations

- When creating a new workspace, you can either automatically create services needed by the workspace or use existing services. If you want to use existing services from a different Azure subscription than the workspace, you must register the Azure Machine Learning namespace in the subscription that contains those services. For example, creating a workspace in subscription A that uses a storage account from subscription B, the Azure Machine Learning namespace must be registered in subscription B before you can use the storage account with the workspace.

The resource provider for Azure Machine Learning is Microsoft.MachineLearningServices. For information on how to see if it is registered and how to register it, see the Azure resource providers and types article.

- By default, creating a workspace also creates an Azure Container Registry (ACR). Since ACR does not currently support unicode characters in resource group names, use a resource group that does not contain these characters.

- Azure Machine Learning does not support hierarchical namespace (Azure Data Lake Storage Gen2 feature) for the workspace's default storage account.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-manage-workspace

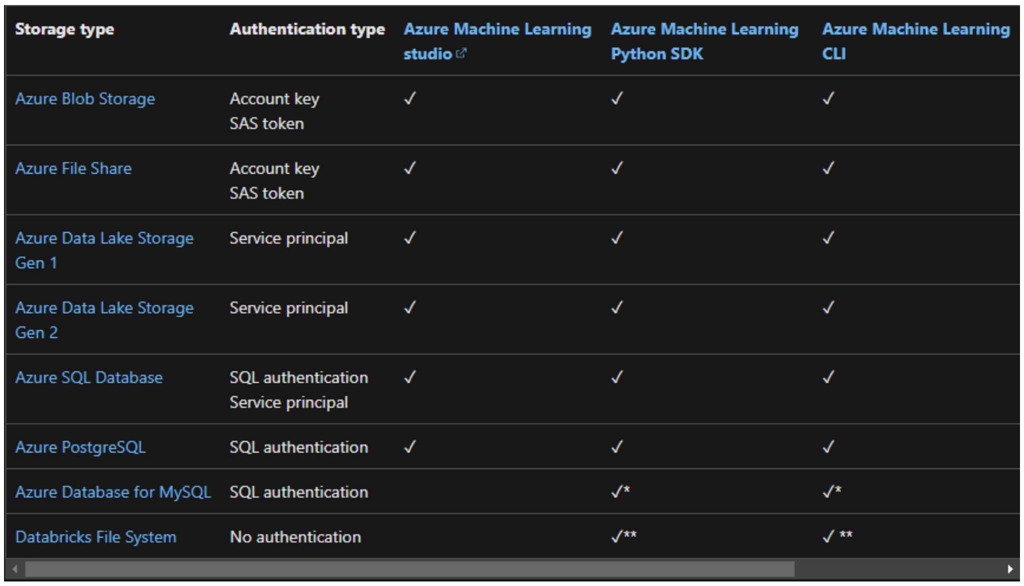
Manage data in an Azure Machine Learning workspace

Select Azure storage resources

Supported data storage service types

Datastores currently support storing connection information to the storage services listed in the following matrix.

* MySQL is only supported for pipeline DataTransferStep
** Databricks is only supported for pipeline DatabricksStep



Storage type	Authentication type	Azure Machine Learning studio	Azure Machine Learning Python SDK	Azure Machine Learning CLI
Azure Blob Storage	Account key SAS token	✓	✓	✓
Azure File Share	Account key SAS token	✓	✓	✓
Azure Data Lake Storage Gen 1	Service principal	✓	✓	✓
Azure Data Lake Storage Gen 2	Service principal	✓	✓	✓
Azure SQL Database	SQL authentication Service principal	✓	✓	✓
Azure PostgreSQL	SQL authentication	✓	✓	✓
Azure Database for MySQL	SQL authentication		✓*	✓*
Databricks File System	No authentication		✓**	✓ **

Storage guidance

We recommend creating a datastore for an Azure Blob container. Both standard and premium storage are available for blobs. Although premium storage is more expensive, its faster throughput speeds might improve the speed of your training runs, particularly if you train against a large dataset. For information about the cost of storage accounts, see the Azure pricing calculator.

Azure Data Lake Storage Gen2 is built on top of Azure Blob storage and designed for enterprise big data analytics. A fundamental part of Data Lake Storage Gen2 is the addition of a hierarchical namespace to Blob storage. The hierarchical namespace organizes objects/files into a hierarchy of directories for efficient data access.

Storage access and permissions

To ensure you securely connect to your Azure storage service, Azure Machine Learning requires that you have permission to access the corresponding data storage container. This access depends on the authentication credentials used to register the datastore.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-access-data

Register and maintain datastores

Datastores securely connect to your storage service on Azure without putting your authentication credentials and the integrity of your original data source at risk. They store connection information, like your subscription ID and token authorization in your Key Vault that's associated with the workspace, so you can securely access your storage without having to hard code them in your scripts. You can create datastores that connect to these Azure storage solutions.

To understand where datastores fit in Azure Machine Learning's overall data access workflow, see the Securely access data article.

Create and register datastores

When you register an Azure storage solution as a datastore, you automatically create and register that datastore to a specific workspace. Review the storage access & permissions section for guidance on virtual network scenarios, and where to find required authentication credentials.

Within this section are examples for how to create and register a datastore via the Python SDK for the following storage types. The parameters provided in these examples are the required parameters to create and register a datastore.

- Azure blob container
- Azure file share
- Azure Data Lake Storage Generation 2

To create datastores for other supported storage services, see the reference documentation for the applicable register_azure_* methods.

If you prefer a low code experience, see Connect to data with Azure Machine Learning studio.

Create datastores with other Azure tools

In addition to creating datastores with the Python SDK and the studio, you can also use Azure Resource Manager templates or the Azure Machine Learning VS Code extension.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-access-data

Create and manage datasets

Create Azure Machine Learning datasets

In this article, you learn how to create Azure Machine Learning datasets to access data for your local or remote experiments with the Azure Machine Learning Python SDK. To understand where datasets fit in Azure Machine Learning's overall data access workflow, see the Securely access data article.

By creating a dataset, you create a reference to the data source location, along with a copy of its metadata. Because the data remains in its existing location, you incur no extra storage cost, and don't risk the integrity of your data sources. Also datasets are lazily evaluated, which aids in workflow performance speeds. You can create datasets from datastores, public URLs, and Azure Open Datasets.

For a low-code experience, Create Azure Machine Learning datasets with the Azure Machine Learning studio.

With Azure Machine Learning datasets, you can:

- Keep a single copy of data in your storage, referenced by datasets.
- Seamlessly access data during model training without worrying about connection strings or data paths. Learn more about how to train with datasets.
- Share data and collaborate with other users

Create datasets from datastores

For the data to be accessible by Azure Machine Learning, datasets must be created from paths in Azure Machine Learning datastores or web URLs.

To create datasets from a datastore with the Python SDK:

1. Verify that you have contributor or owner access to the underlying storage service of your registered Azure Machine Learning datastore. Check your storage account permissions in the Azure portal.

2. Create the dataset by referencing paths in the datastore. You can create a dataset from multiple paths in multiple datastores. There is no hard limit on the number of files or data size that you can create a dataset from.

Create a dataset from pandas dataframe

To create a TabularDataset from an in memory pandas dataframe use the register_pandas_dataframe() method. This method registers the TabularDataset to the workspace and uploads data to your underlying storage, which incurs storage costs.

Create datasets using Azure Resource Manager

There are many templates at https://github.com/Azure/azure-quickstart-templates/tree/master//quickstarts/microsoft.machinelearningservices that can be used to create datasets.

For information on using these templates, see Use an Azure Resource Manager template to create a workspace for Azure Machine Learning.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-register-datasets

Manage compute for experiments in Azure Machine Learning

Determine the appropriate compute specifications for a training workload

What are compute targets in Azure Machine Learning?

A *compute target* is a designated compute resource or environment where you run your training script or host your service deployment. This location might be your local machine or a cloud-based compute resource. Using compute targets makes it easy for you to later change your compute environment without having to change your code.

In a typical model development lifecycle, you might:

1. Start by developing and experimenting on a small amount of data. At this stage, use your local environment, such as a local computer or cloud-based virtual machine (VM), as your compute target.
2. Scale up to larger data, or do distributed training by using one of these training compute targets.
3. After your model is ready, deploy it to a web hosting environment with one of these deployment compute targets.

The compute resources you use for your compute targets are attached to a workspace. Compute resources other than the local machine are shared by users of the workspace.

Training compute targets

Azure Machine Learning has varying support across different compute targets. A typical model development lifecycle starts with development or experimentation on a small amount of data. At this stage, use a local environment like your local computer or a cloud-based VM. As you scale up your training on larger datasets or perform distributed training, use Azure Machine Learning compute to create a single- or multi-node cluster that autoscales each time you submit a run. You can also attach your own compute resource, although support for different scenarios might vary.

Compute targets can be reused from one training job to the next. For example, after you attach a remote VM to your workspace, you can reuse it for multiple jobs. For machine learning pipelines, use the appropriate pipeline step for each compute target.

You can use any of the following resources for a training compute target for most jobs. Not all resources can be used for automated machine learning, machine learning pipelines, or designer. Azure Databricks can be used as a training resource for local runs and machine learning pipelines, but not as a remote target for other training.

Compute targets for inference

When performing inference, Azure Machine Learning creates a Docker container that hosts the model and associated resources needed to use it. This container is then used in a compute target.

The compute target you use to host your model will affect the cost and availability of your deployed endpoint. Use this table to choose an appropriate compute target.

Supported VM series and sizes
When you select a node size for a managed compute resource in Azure Machine Learning, you can choose from among select VM sizes available in Azure. Azure offers a range of sizes for Linux and Windows for different workloads. To learn more, see VM types and sizes.

There are a few exceptions and limitations to choosing a VM size:

- Some VM series aren't supported in Azure Machine Learning.
- Some VM series are restricted. To use a restricted series, contact support and request a quota increase for the series.

https://docs.microsoft.com/en-us/azure/machine-learning/concept-compute-target

Create compute targets for experiments and training

What's a compute target?

With Azure Machine Learning, you can train your model on various resources or environments, collectively referred to as **compute targets**. A compute target can be a local machine or a cloud resource, such as an Azure Machine Learning Compute, Azure HDInsight, or a remote virtual machine. You also use compute targets for model deployment as described in "Where and how to deploy your models".

Azure Machine Learning also supports attaching an Azure Virtual Machine. The VM must be an Azure Data Science Virtual Machine (DSVM). The VM offers a curated choice of tools and frameworks for full-lifecycle machine learning development

Create: Azure Machine Learning cannot create a remote VM for you. Instead, you must create the VM and then attach it to your Azure Machine Learning workspace. For information on creating a DSVM, see Provision the Data Science Virtual Machine for Linux (Ubuntu).

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-attach-compute-targets

Configure Attached Compute resources including Azure Databricks

Learn how to attach Azure compute resources to your Azure Machine Learning workspace. Then you can use these resources as training and inference compute targets in your machine learning tasks.

In this article, learn how to set up your workspace to use these compute resources:

- Your local computer
- Remote virtual machines
- Apache Spark pools (powered by Azure Synapse Analytics)
- Azure HDInsight
- Azure Batch
- Azure Databricks - used as a training compute target only in machine learning pipelines
- Azure Data Lake Analytics
- Azure Container Instance
- Azure Kubernetes Service & Azure Arc-enabled Kubernetes (preview)

To use compute targets managed by Azure Machine Learning, see:

- Azure Machine Learning compute instance
- Azure Machine Learning compute cluster
- Azure Kubernetes Service cluster

Azure Databricks

Azure Databricks is an Apache Spark-based environment in the Azure cloud. It can be used as a compute target with an Azure Machine Learning pipeline.
To attach Azure Databricks as a compute target, provide the following information:

- Databricks compute name: The name you want to assign to this compute resource.
- Databricks workspace name: The name of the Azure Databricks workspace.
- Databricks access token: The access token used to authenticate to Azure Databricks. To generate an access token, see the Authentication document.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-attach-compute-targets

Monitor compute utilization

When you have critical applications and business processes relying on Azure resources, you want to monitor those resources for their availability, performance, and operation. This article describes the monitoring data generated by Azure Machine Learning and how to analyze and alert on this data with Azure Monitor.

The information in this document is primarily for administrators, as it describes monitoring for the Azure Machine Learning service and associated Azure services. If you are a data scientist or developer, and want to monitor information specific to your *model training runs*, see the following documents:

- Start, monitor, and cancel training runs
- Log metrics for training runs
- Track experiments with MLflow
- Visualize runs with TensorBoard

If you want to monitor information generated by models deployed as web services, see Collect model data and Monitor with Application Insights.

https://docs.microsoft.com/en-us/azure/machine-learning/monitor-azure-machine-learning

Implement security and access control in Azure Machine Learning

Determine access requirements and map requirements to built-in roles

Manage access to an Azure Machine Learning workspace

In this article, you learn how to manage access (authorization) to an Azure Machine Learning workspace. Azure role-based access control (Azure RBAC) is used to manage access to Azure resources, such as the ability to create new resources or use existing ones. Users in your Azure Active Directory (Azure AD) are assigned specific roles, which grant access to resources. Azure provides both built-in roles and the ability to create custom roles.

While this article focuses on Azure Machine Learning, individual services that Azure ML relies on provide their own RBAC settings. For example, using the information in this article, you can configure who can submit scoring requests to a model deployed as a web service on Azure Kubernetes Service. But Azure Kubernetes Service provides its own set of Azure roles. For service specific RBAC information that may be useful with Azure Machine Learning, see the following links:

- Control access to Azure Kubernetes cluster resources
- Use Azure RBAC for Kubernetes authorization
- Use Azure RBAC for access to blob data

Applying some roles may limit UI functionality in Azure Machine Learning studio for other users. For example, if a user's role does not have the ability to create a compute instance, the option to create a compute instance will not be available in studio. This behavior is expected, and prevents the user from attempting operations that would return an access denied error.

Default roles

Azure Machine Learning workspaces have a four built-in roles that are available by default. When adding users to a workspace, they can be assigned one of the built-in roles described below.

Manage workspace access

If you're an owner of a workspace, you can add and remove roles for the workspace. You can also assign roles to users. Use the following links to discover how to manage access:

- Azure portal UI
- PowerShell
- Azure CLI
- REST API
- Azure Resource Manager templates

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-assign-roles

Create custom roles

If the built-in roles are insufficient, you can create custom roles. Custom roles might have read, write, delete, and compute resource permissions in that workspace. You can make the role available at a specific workspace level, a specific resource group level, or a specific subscription level.

To create a custom role, first construct a role definition JSON file that specifies the permission and scope for the role. The following example defines a custom role named "Data Scientist Custom" scoped at a specific workspace level

This custom role can do everything in the workspace except for the following actions:

- It can't create or update a compute resource.
- It can't delete a compute resource.
- It can't add, delete, or alter role assignments.
- It can't delete the workspace.

After deployment, this role becomes available in the specified workspace. Now you can add and assign this role in the Azure portal.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-assign-roles

Manage role membership

Azure built-in roles

Azure role-based access control (Azure RBAC) has several Azure built-in roles that you can assign to users, groups, service principals, and managed identities. Role assignments are the way you control access to Azure resources. If the built-in roles don't meet the specific needs of your organization, you can create your own Azure custom roles. For information about how to assign roles, see Steps to assign an Azure role.

This article lists the Azure built-in roles. If you are looking for administrator roles for Azure Active Directory (Azure AD), see Azure AD built-in roles.

The following table provides a brief description of each built-in role. Click the role name to see the list of Actions, NotActions, DataActions, and NotDataActions for each role. For information about what these actions mean and how they apply to the control and data planes, see Understand Azure role definitions.

https://docs.microsoft.com/en-us/azure/role-based-access-control/built-in-roles

Manage credentials by using Azure Key Vault

Enterprise security and governance for Azure Machine Learning

In this article, you'll learn about security and governance features available for Azure Machine Learning. These features are useful for administrators, DevOps, and MLOps who want to create a secure configuration that is compliant with your companies policies. With Azure Machine Learning and the Azure platform, you can:

- Restrict access to resources and operations by user account or groups
- Restrict incoming and outgoing network communications
- Encrypt data in transit and at rest
- Scan for vulnerabilities
- Apply and audit configuration policies

Restrict access to resources and operations

Azure Active Directory (Azure AD) is the identity service provider for Azure Machine Learning. It allows you to create and manage the security objects (user, group, service principal, and managed identity) that are used to *authenticate* to Azure resources. Multi-factor authentication is supported if Azure AD is configured to use it.

Here's the authentication process for Azure Machine Learning using multi-factor authentication in Azure AD:

1. The client signs in to Azure AD and gets an Azure Resource Manager token.
2. The client presents the token to Azure Resource Manager and to all Azure Machine Learning.
3. Azure Machine Learning provides a Machine Learning service token to the user compute target (for example, Azure Machine Learning compute cluster). This token is used by the user compute target to call back into the Machine Learning service after the run is complete. The scope is limited to the workspace.

Each workspace has an associated system-assigned managed identity that has the same name as the workspace. This managed identity is used to securely access resources used by the workspace. It has the following Azure RBAC permissions on associated resources:

The system-assigned managed identity is used for internal service-to-service authentication between Azure Machine Learning and other Azure resources. The identity token is not accessible to users and cannot be used by them to gain access to these resources. Users can only access the resources through Azure Machine Learning control and data plane APIs, if they have sufficient RBAC permissions.

The managed identity needs Contributor permissions on the resource group containing the workspace in order to provision the associated resources, and to deploy Azure Container Instances for web service endpoints.

We don't recommend that admins revoke the access of the managed identity to the resources mentioned in the preceding table. You can restore access by using the resync keys operation.

You can provision the workspace to use user-assigned managed identity, and grant the managed identity additional roles, for example to access your own Azure Container Registry for base Docker images. For more information, see Use managed identities for access control.

You can also configure managed identities for use with Azure Machine Learning compute cluster. This managed identity is independent of workspace managed identity. With a compute cluster, the managed identity is used to access resources such as secured datastores that the user running the training job may not have access to. For more information, see Identity-based data access to storage services on Azure.

https://docs.microsoft.com/en-us/azure/machine-learning/concept-enterprise-security

Set up an Azure Machine Learning development environment

Create compute instances

Create and manage an Azure Machine Learning compute instance

Learn how to create and manage a compute instance in your Azure Machine Learning workspace.

Use a compute instance as your fully configured and managed development environment in the cloud. For development and testing, you can also use the instance as a training compute target or for an inference target. A compute instance can run multiple jobs in parallel and has a job queue. As a development environment, a compute instance cannot be shared with other users in your workspace.

In this article, you learn how to:

- Create a compute instance
- Manage (start, stop, restart, delete) a compute instance
- Create a schedule to automatically start and stop the compute instance (preview)
- Use a setup script to customize and configure the compute instance

Compute instances can run jobs securely in a virtual network environment, without requiring enterprises to open up SSH ports. The job executes in a containerized environment and packages your model dependencies in a Docker container.

Create

Creating a compute instance is a one time process for your workspace. You can reuse the compute as a development workstation or as a compute target for training. You can have multiple compute instances attached to your workspace.

The dedicated cores per region per VM family quota and total regional quota, which applies to compute instance creation, is unified and shared with Azure Machine Learning training compute cluster quota. Stopping the compute instance does not release quota to ensure you will be able to restart the compute instance. It is not possible to change the virtual machine size of compute instance once it is created.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-manage-compute-instance

Share compute instances

Customize the compute instance with a script (preview)

Use a setup script for an automated way to customize and configure the compute instance at provisioning time. As an administrator, you can write a customization script to be used to provision all compute instances in the workspace according to your requirements.

Some examples of what you can do in a setup script:

- Install packages, tools, and software
- Mount data
- Create custom conda environment and Jupyter kernels
- Clone git repositories and set git config
- Set network proxies
- Set environment variables
- Install JupyterLab extensions

Manage

Start, stop, restart, and delete a compute instance. A compute instance does not automatically scale down, so make sure to stop the resource to prevent ongoing charges. Stopping a compute instance deallocates it. Then start it again when you need it. While stopping the compute instance stops the billing for compute hours, you will still be billed for disk, public IP, and standard load balancer.

You can create a schedule for the compute instance to automatically start and stop based on a time and day of week.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-manage-compute-instance

Access Azure Machine Learning workspaces from other development environments

This article describes how to use your local computer as a target for training or deploying models created in Azure Machine Learning. Azure Machine Learning is flexible enough to work with most Python machine learning frameworks. Machine learning solutions generally have complex dependencies that can be difficult to duplicate. This article will show you how to balance total control with ease of use.

Scenarios for local deployment include:

- Quickly iterating data, scripts, and models early in a project.
- Debugging and troubleshooting in later stages.
- Final deployment on user-managed hardware.

Prepare your local machine

The most reliable way to locally run an Azure Machine Learning model is with a Docker image. A Docker image provides an isolated, containerized experience that duplicates, except for hardware issues, the Azure execution environment. For more information on installing and configuring Docker for development scenarios, see Overview of Docker remote development on Windows.

It's possible to attach a debugger to a process running in Docker. (See Attach to a running container.) But you might prefer to debug and iterate your Python code without involving Docker. In this scenario, it's important that your local machine uses the same libraries that are used when you run your experiment in Azure Machine Learning. To manage Python dependencies, Azure uses conda. You can re-create the environment by using other package managers, but installing and configuring conda on your local machine is the easiest way to synchronize.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-local?WT.mc_id=AI-MVP-5004069

Set up an Azure Databricks workspace

Create an Azure Databricks workspace

Quickstart: Run a Spark job on Azure Databricks Workspace using the Azure portal

In this quickstart, you use the Azure portal to create an Azure Databricks workspace with an Apache Spark cluster. You run a job on the cluster and use custom charts to produce real-time reports from Seattle safety data.

Create an Azure Databricks workspace

In this section, you create an Azure Databricks workspace using the Azure portal or the Azure CLI.

1. In the Azure portal, select Create a resource > Analytics > Azure Databricks.
2. Under Azure Databricks Service, provide the values to create a Databricks workspace.
3. Select Review + Create, and then Create. The workspace creation takes a few minutes. During workspace creation, you can view the deployment status in Notifications. Once this process is finished, your user account is automatically added as an admin user in the workspace.

   When a workspace deployment fails, the workspace is still created in a failed state. Delete the failed workspace and create a new workspace that resolves the deployment errors. When you delete the failed workspace, the managed resource group and any successfully deployed resources are also deleted.

Create a Spark cluster in Databricks

In the Azure portal, go to the Databricks workspace that you created, and then click Launch Workspace.

You are redirected to the Azure Databricks portal. From the portal, click New Cluster.

In the New cluster page, provide the values to create a cluster.

- Accept all other default values other than the following:
- Enter a name for the cluster.
- For this article, create a cluster with (5.X, 6.X, 7.X) runtime.
- Make sure you select the Terminate after _ minutes of inactivity checkbox. Provide a

  duration (in minutes) to terminate the cluster, if the cluster is not being used.

  Select Create cluster. Once the cluster is running, you can attach notebooks to the cluster and run Spark jobs.

https://docs.microsoft.com/en-us/azure/databricks/scenarios/quickstart-create-databricks-workspace-portal

Create an Azure Databricks cluster

Create a cluster

There are two types of clusters:

- *All-Purpose clusters* can be shared by multiple users. These are typically used to run notebooks. All-Purpose clusters remain active until you terminate them.
- *Job clusters* run a job. You create a job cluster when you create a job. Such clusters are terminated automatically after the job is completed.

This article describes how to create an all-purpose cluster. To learn how to create job clusters, see Create a job.

Use the Create button

The easiest way to create a new cluster is to use the **Create** button:

1. Click ⊕ Create in the sidebar and select Cluster from the menu. The Create Cluster page appears.

2. Name and configure the cluster.

   1.There are many cluster configuration options, which are described in detail in cluster configuration.

3. Click the Create Cluster button.

The cluster Configuration tab displays a spinning progress indicator while the cluster is in a pending state. When the cluster has started and is ready to use, the progress spinner turns into a green circle with a check mark. This indicates that cluster is in the running state, and you can now attach notebooks and start running commands and queries

https://docs.microsoft.com/en-us/azure/databricks/clusters/create

Create and run notebooks in Azure Databricks

You can manage notebooks using the UI, the CLI, and by invoking the Workspace API. This article focuses on performing notebook tasks using the UI. For the other methods, see Databricks CLI and Workspace API 2.0.

Create a notebook

Use the Create button
The easiest way to create a new notebook in your default folder is to use the Create button:

1. Click Create in the sidebar and select Notebook from the menu. The Create Notebook dialog appears.
2. Enter a name and select the notebook's default language.
3. If there are running clusters, the Cluster drop-down displays. Select the cluster you want to attach the notebook to.
4. Click Create.

Create a notebook in any folder

You can create a new notebook in any folder (for example, in the **Shared** folder) following these steps:

1. In the sidebar, click Workspace. Do one of the following:

- Next to any folder, click the on the right side of the text and select Create > Notebook.
- In the workspace or a user folder, click and select Create > Notebook.

2. Follow steps 2 through 4 in Use the Create button.

https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage

Link and Azure Databricks workspace to an Azure Machine Learning workspace

Set up a development environment with Azure Databricks and AutoML in Azure Machine Learning

Learn how to configure a development environment in Azure Machine Learning that uses Azure Databricks and automated ML.

Azure Databricks is ideal for running large-scale intensive machine learning workflows on the scalable Apache Spark platform in the Azure cloud. It provides a collaborative Notebook-based environment with a CPU or GPU-based compute cluster.

For information on other machine learning development environments, see Set up Python development environment.

Prerequisite

Azure Machine Learning workspace. If you don't have one, you can create an Azure Machine Learning workspace through the Azure portal, Azure CLI, and Azure Resource Manager templates.

Azure Databricks with Azure Machine Learning and AutoML

Azure Databricks integrates with Azure Machine Learning and its AutoML capabilities.

You can use Azure Databricks:

- To train a model using Spark MLlib and deploy the model to ACI/AKS.
- With automated machine learning capabilities using an Azure ML SDK.
- As a compute target from an Azure Machine Learning pipeline.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-databricks-automl-environment

Run experiments and train models (20-25%)

Create models by using the Azure Machine Learning designer

Create a training pipeline by using Azure Machine Learning designer

Create a new pipeline

Azure Machine Learning pipelines organize multiple machine learning and data processing steps into a single resource. Pipelines let you organize, manage, and reuse complex machine learning workflows across projects and users.

To create an Azure Machine Learning pipeline, you need an Azure Machine Learning workspace. In this section, you learn how to create both these resources.

Create a new workspace

You need an Azure Machine Learning workspace to use the designer. The workspace is the top -level resource for Azure Machine Learning, it provides a centralized place to work with all the artifacts you create in Azure Machine Learning. For instruction on creating a workspace, see Create and manage Azure Machine Learning workspaces.

Create the pipeline

1. Sign in to ml.azure.com, and select the workspace you want to work with.
2. Select Designer.
3. Select Easy-to-use prebuilt components.
4. At the top of the canvas, select the default pipeline name Pipeline-Created-on. Rename it to *Automobile price prediction*. The name doesn't need to be unique.

https://docs.microsoft.com/en-us/azure/machine-learning/tutorial-designer-automobile-price-train-score

Ingest data in a designer pipeline

Import data
There are several sample datasets included in the designer for you to experiment with. For this tutorial, use Automobile price data (Raw).

1. To the left of the pipeline canvas is a palette of datasets and components. Select Sample datasets to view the available sample datasets.

2. Select the dataset Automobile price data (Raw), and drag it onto the canvas.

Visualize the data
You can visualize the data to understand the dataset that you'll use.

1. Right-click the Automobile price data (Raw) and select Preview Data.

2. Select the different columns in the data window to view information about each one.

   Each row represents an automobile, and the variables associated with each automobile appear as columns. There are 205 rows and 26 columns in this dataset.

Prepare data
Datasets typically require some preprocessing before analysis. You might have noticed some missing values when you inspected the dataset. These missing values must be cleaned so that the model can analyze the data correctly.

https://docs.microsoft.com/en-us/azure/machine-learning/tutorial-designer-automobile-price-train-score

Use designer modules to define a pipeline data flow

Use the Evaluate Model component to evaluate how well your model scored the test dataset.

1. Enter evaluate in the search box to find the Evaluate Model component. Drag the component to the pipeline canvas.
2. Connect the output of the Score Model component to the left input of Evaluate Model.

https://docs.microsoft.com/en-us/azure/machine-learning/tutorial-designer-automobile-price-train-score

Use custom code modules in designer

Create Python Model component

This article describes a component in Azure Machine Learning designer.

Learn how to use the Create Python Model component to create an untrained model from a Python script. You can base the model on any learner that's included in a Python package in the Azure Machine Learning designer environment.

After you create the model, you can use Train Model to train the model on a dataset, like any other learner in Azure Machine Learning. The trained model can be passed to Score Model to make predictions. You can then save the trained model and publish the scoring workflow as a web service.t

Currently, it's not possible to connect this component to Tune Model Hyperparameters component or pass the scored results of a Python model to Evaluate Model. If you need to tune the hyperparameters or evaluate a model, you can write a custom Python script by using Execute Python Script component.

Configure the component

Use of this component requires intermediate or expert knowledge of Python. The component supports use of any learner that's included in the Python packages already installed in Azure Machine Learning. See the preinstalled Python package list in Execute Python Script.

This article shows how to use Create Python Model with a simple pipeline. Here's a diagram of the pipeline:

Select Create Python Model, and edit the script to implement your modeling or data management process. You can base the model on any learner that's included in a Python package in the Azure Machine Learning environment.

Connect the Create Python Model component that you just created to Train Model and Score Model.

If you need to evaluate the model, add an Execute Python Script component and edit the Python script.

https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/create-python-model

Run model training scripts

Create and run an experiment by using the Azure Machine Learning SDK

Quickstart: Create workspace resources you need to get started with Azure Machine Learning

In this quickstart, you'll create a workspace and then add compute resources to the workspace. You'll then have everything you need to get started with Azure Machine Learning.

The workspace is the top-level resource for your machine learning activities, providing a centralized place to view and manage the artifacts you create when you use Azure Machine Learning. The compute resources provide a pre-configured cloud-based environment you can use to train, deploy, automate, manage, and track machine learning models.

Create the workspace

If you already have a workspace, skip this section and continue to Create a compute instance.

If you don't yet have a workspace, create one now:

1. Sign in to the Azure portal by using the credentials for your Azure subscription.
2. In the upper-left corner of the Azure portal, select the three bars, then + Create a resource.
3. Use the search bar to find Machine Learning.
4. Select Machine Learning.
5. In the Machine Learning pane, select Create to begin.
6. Provide the following information to configure your new workspace:
7. After you're finished configuring the workspace, select Review + Create.
8. Select Create to create the workspace.

   When the process is finished, a deployment success message appears.
9. To view the new workspace, select Go to resource.
10. From the portal view of your workspace, select Launch studio to go to the Azure Machine Learning studio.

https://docs.microsoft.com/en-us/azure/machine-learning/quickstart-create-resources

Configure run settings for a script

Configure and submit training runs

In this article, you learn how to configure and submit Azure Machine Learning runs to train your models. Snippets of code explain the key parts of configuration and submission of a training script. Then use one of the example notebooks to find the full end-to-end working examples.

When training, it is common to start on your local computer, and then later scale out to a cloud-based cluster. With Azure Machine Learning, you can run your script on various compute targets without having to change your training script.

All you need to do is define the environment for each compute target within a **script run** configuration. Then, when you want to run your training experiment on a different compute target, specify the run configuration for that compute.

What's a script run configuration?

A ScriptRunConfig is used to configure the information necessary for submitting a training run as part of an experiment.

You submit your training experiment with a ScriptRunConfig object. This object includes the:

- source_directory: The source directory that contains your training script
- script: The training script to run
- compute_target: The compute target to run on
- environment: The environment to use when running the script
- and some additional configurable options (see the reference documentation for more information)

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-set-up-training-targets

Consume data from a dataset in an experiment by using the Azure Machine Learning SDK

Datastores securely connect to your storage service on Azure without putting your authentication credentials and the integrity of your original data source at risk. They store connection information, like your subscription ID and token authorization in your Key Vault that's associated with the workspace, so you can securely access your storage without having to hard code them in your scripts. You can create datastores that connect to these Azure storage solutions.

Storage guidance

We recommend creating a datastore for an Azure Blob container. Both standard and premium storage are available for blobs. Although premium storage is more expensive, its faster throughput speeds might improve the speed of your training runs, particularly if you train against a large dataset. For information about the cost of storage accounts, see the Azure pricing calculator.

Azure Data Lake Storage Gen2 is built on top of Azure Blob storage and designed for enterprise big data analytics. A fundamental part of Data Lake Storage Gen2 is the addition of a hierarchical namespace to Blob storage. The hierarchical namespace organizes objects/files into a hierarchy of directories for efficient data access.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-access-data

Run a training script on Azure Databricks compute / Run code to train a model in an Azure Databricks notebook

scikit-learn notebook

Notebook	Requirements	Features
Machine learning quickstart	Databricks Runtime 7.5 ML or above	Classification model, MLflow, automated hyperparameter tuning with Hyperopt and MLflow
Machine learning with Model Registry	Databricks Runtime 7.1 ML or above	Classification model, MLflow, automated hyperparameter tuning with Hyperopt and MLflow, Model Registry
End-to-end example	Databricks Runtime 6.5 ML or above	Classification model, MLflow, automated hyperparameter tuning with Hyperopt and MLflow, XGBoost, Model Registry, Model Serving

Apache Spark MLlib notebook

Notebook	Requirements	Features
Machine learning with MLlib	Databricks Runtime 5.5 LTS ML or above	Logistic regression model, Spark pipeline, automated hyperparameter tuning using MLlib API

Deep learning notebook

Notebook	Requirements	Features
Deep learning with TensorFlow Keras	Databricks Runtime 7.0 ML or above	Neural network model, inline TensorBoard, automated hyperparameter tuning with Hyperopt and MLflow, autologging, ModelRegistry

https://docs.microsoft.com/en-us/azure/databricks/applications/machine-learning/tutorial/

Generate metrics from an experiment run

Log metrics from an experiment run

Log real-time information using both the default Python logging package and Azure Machine Learning Python SDK-specific functionality. You can log locally and send logs to your workspace in the portal.

Logs can help you diagnose errors and warnings, or track performance metrics like parameters and model performance. In this article, you learn how to enable logging in the following scenarios:

- ✓ Log run metrics
- ✓ Interactive training sessions
- ✓ Submitting training jobs using ScriptRunConfig
- ✓ Python native logging settings
- ✓ Logging from additional sources

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-log-view-metrics

Retrieve and view experiment outputs

The Azure Machine Learning SDK for Python, Machine Learning CLI, and Azure Machine Learning studio provide various methods to monitor, organize, and track your runs for training and experimentation. Your ML run history is an important part of an explainable and repeatable ML development process.

The run display name is an optional and customizable name that you can provide for your run. To edit the run display name:

1. Navigate to the runs list.
2. Select the run to edit the display name in the run details page.
3. Select the Edit button to edit the run display name.

Custom View

To view your runs in the studio:

1. Navigate to the **Experiments** tab.

2. Select either **All experiments** to view all the runs in an experiment or select **All runs** to view all the runs submitted in the Workspace.

In the **All runs'** page, you can filter the runs list by tags, experiments, compute target and more to better organize and scope your work.

1. Make customizations to the page by selecting runs to compare, adding charts or applying filters. These changes can be saved as a **Custom View** so you can easily return to your work. Users with workspace permissions can edit, or view the custom view. Also, share the custom view with team members for enhanced collaboration by selecting **Share view**.

2. To view the run logs, select a specific run and in the **Outputs + logs** tab, you can find diagnostic and error logs for your run.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-track-monitor-analyze-runs

Use logs to troubleshoot experiment run errors

Run description

A run description can be added to a run to provide more context and information to the run. You can also search on these descriptions from the runs list and add the run description as a column in the runs list.

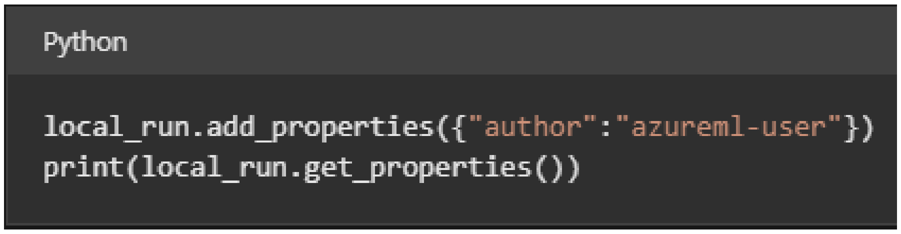
Navigate to the **Run Details** page for your run and select the edit or pencil icon to add, edit, or delete descriptions for your run. To persist the changes to the runs list, save the changes to your existing Custom View or a new Custom View. Markdown format is supported for run descriptions, which allows images to be embedded and deep linking as shown below.

Tag and find runs

In Azure Machine Learning, you can use properties and tags to help organize and query your runs for important information.

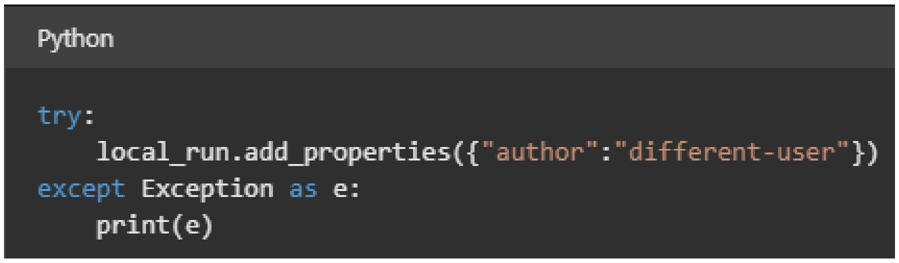
- Add properties and tags

To add searchable metadata to your runs, use the add_properties() method. For example, the following code adds the "author" property to the run:



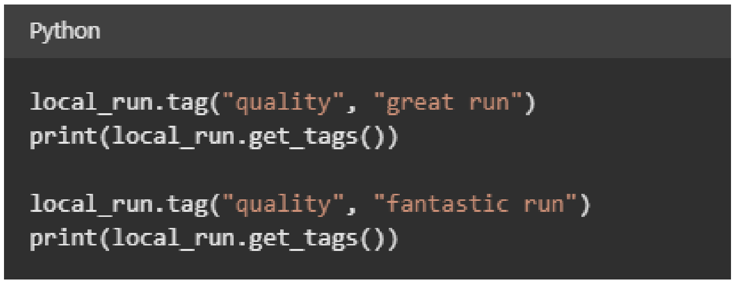
```python
local_run.add_properties({"author":"azureml-user"})
print(local_run.get_properties())
```

Properties are immutable, so they create a permanent record for auditing purposes. The following code example results in an error, because we already added "azureml-user" as the "author" property value in the preceding code:



```python
try:
    local_run.add_properties({"author":"different-user"})
except Exception as e:
    print(e)
```

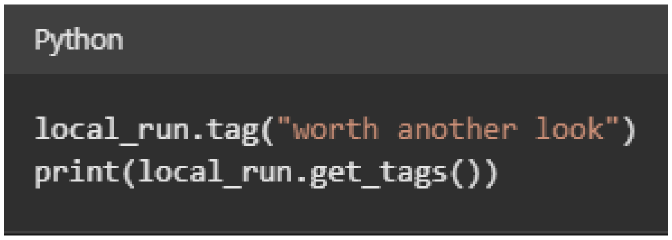
Unlike properties, tags are mutable. To add searchable and meaningful information for consumers of your experiment, use the tag() method.



Python

```python
local_run.tag("quality", "great run")
print(local_run.get_tags())

local_run.tag("quality", "fantastic run")
print(local_run.get_tags())
```

You can also add simple string tags. When these tags appear in the tag dictionary as keys, they have a value of None.



Python

```python
local_run.tag("worth another look")
print(local_run.get_tags())
```

Query properties and tags
You can query runs within an experiment to return a list of runs that match specific properties and tags.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-track-monitor-analyze-runs

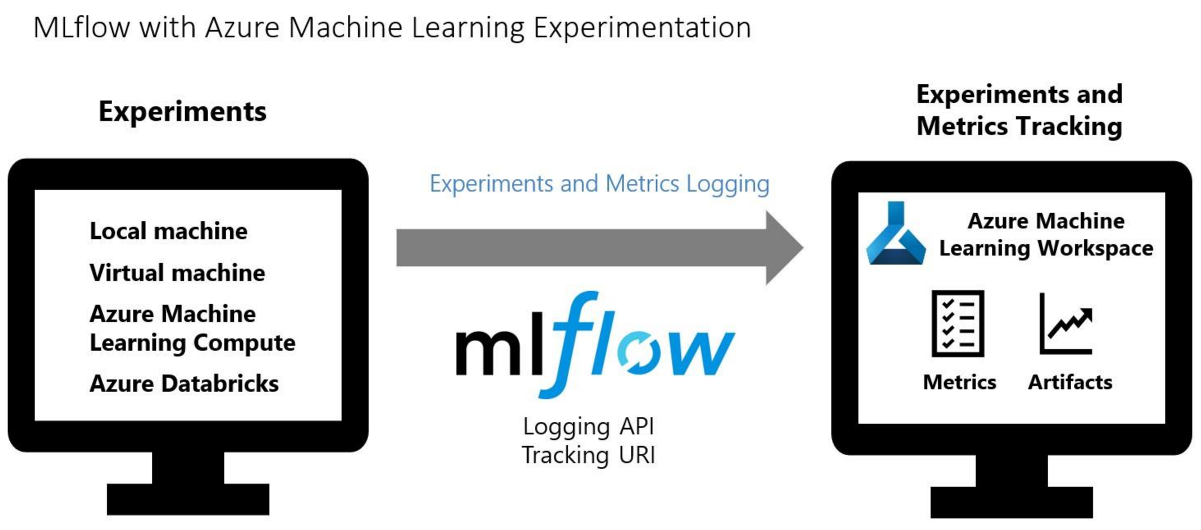
Use MLflow to track experiments

Supported capabilities include:

- Track and log experiment metrics and artifacts in your Azure Machine Learning workspace. If you already use MLflow Tracking for your experiments, the workspace provides a centralized, secure, and scalable location to store training metrics and models.
- Submit training jobs with MLflow Projects with Azure Machine Learning backend support (preview). You can submit jobs locally with Azure Machine Learning tracking or migrate your runs to the cloud like via an Azure Machine Learning Compute.

MLflow is an open-source library for managing the life cycle of your machine learning experiments. MLflow Tracking is a component of MLflow that logs and tracks your training run metrics and model artifacts, no matter your experiment's environment--locally on your computer, on a remote compute target, a virtual machine, or an Azure Databricks cluster.

The following diagram illustrates that with MLflow Tracking, you track an experiment's run metrics and store model artifacts in your Azure Machine Learning workspace.


MLflow with Azure Machine Learning Experimentation
Experiments
Local machine
Virtual machine
Azure Machine Learning Compute
Azure Databricks
Experiments and Metrics Logging
mlflow
Logging API
Tracking URI
Experiments and Metrics Tracking
Azure Machine Learning Workspace
Metrics
Artifacts

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-use-mlflow

Track experiments running in Azure Databricks

The MLflow tracking component lets you log source properties, parameters, metrics, tags, and artifacts related to training a machine learning model. To get started with MLflow, try one of the MLflow quickstart tutorials.

MLflow tracking is based on two concepts, *experiments* and *runs*:

- An MLflow *experiment* is the primary unit of organization and access control for MLflow runs; all MLflow runs belong to an experiment. Experiments let you visualize, search for, and compare runs, as well as download run artifacts and metadata for analysis in other tools.
- An MLflow *run* corresponds to a single execution of model code. Each run records the following information:
- **Source**: Name of the notebook that launched the run or the project name and entry point for the run.
- **Version**: Notebook revision if run from a notebook or Git commit hash if run from an MLflow Project.
- **Start & end time**: Start and end time of the run.
- **Parameters**: Model parameters saved as key-value pairs. Both keys and values are strings.
- **Metrics**: Model evaluation metrics saved as key-value pairs. The value is numeric. Each metric can be updated throughout the course of the run (for example, to track how your model's loss function is converging), and MLflow records and lets you visualize the metric's history.
- **Tags**: Run metadata saved as key-value pairs. You can update tags during and after a run completes. Both keys and values are strings.
- **Artifacts**: Output files in any format. For example, you can record images, models (for example, a pickled scikit-learn model), and data files (for example, a Parquet file) as an artifact.

The MLflow Tracking API logs parameters, metrics, tags, and artifacts from a model run. The Tracking API communicates with an MLflow tracking server. When you use Databricks, a Databricks-hosted tracking server logs the data. The hosted MLflow tracking server has Python, Java, and R APIs.

https://docs.microsoft.com/en-us/azure/databricks/applications/mlflow/tracking

Use Automated Machine Learning to create optimal models

Use the Automated ML interface in Azure Machine Learning studio

Automated machine learning, AutoML, is a process in which the best machine learning algorithm to use for your specific data is selected for you. This process enables you to generate machine learning models quickly

Create and run experiment

1. Select **+ New automated ML run** and populate the form.
2. Select a dataset from your storage container, or create a new dataset. Datasets can be created from local files, web urls, datastores, or Azure open datasets. Learn more about dataset creation.
3. Select your newly created dataset once it appears. You are also able to view a preview of the dataset and sample statistics.
4. On the Configure run form, select Create new and enter Tutorial-automl-deploy for the experiment name.
5. Select a target column; this is the column that you would like to do predictions on.
6. Select a compute type for the data profiling and training job. You can select a compute cluster or compute instance.
7. Select a compute from the dropdown list of your existing computes. To create a new compute, follow the instructions in step 8.
8. Select Create a new compute to configure your compute context for this experiment.
9. On the Task type and settings form, select the task type: classification, regression, or forecasting. See supported task types for more information.
10. (Optional) View addition configuration settings: additional settings you can use to better control the training job. Otherwise, defaults are applied based on experiment selection and data.
11. (Optional) View featurization settings: if you choose to enable Automatic featurization in the Additional configuration settings form, default featurization techniques are applied. In the View featurization settings you can change these defaults and customize accordingly. Learn how to customize featurizations.
12. The [Optional] Validate and test form allows you to do the following.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-use-automated-ml-for-ml-models

Use Automated ML from the Azure Machine Learning SDK

Automated machine learning supports data that resides on your local desktop or in the cloud such as Azure Blob Storage. The data can be read into a Pandas DataFrame or an Azure Machine Learning TabularDataset. Learn more about datasets.

Requirements for training data in machine learning:

- Data must be in tabular form.
- The value to predict, target column, must be in the data.

For remote experiments, training data must be accessible from the remote compute. Automated ML only accepts Azure Machine Learning TabularDatasets when working on a remote compute.
Azure Machine Learning datasets expose functionality to:

- Easily transfer data from static files or URL sources into your workspace.
- Make your data available to training scripts when running on cloud compute resources.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-train

Select pre-processing options

FeaturizationConfig Indicator for whether featurization step should be done automatically or not, or whether customized featurization should be used. Note: If the input data is sparse, featurization cannot be turned on.
Column type is automatically detected. Based on the detected column type preprocessing/ featurization is done as follows:

- Categorical: Target encoding, one hot encoding, drop high cardinality categories, impute missing values.
- Numeric: Impute missing values, cluster distance, weight of evidence.
- DateTime: Several features such as day, seconds, minutes, hours etc.
- Text: Bag of words, pre-trained Word embedding, text target encoding.

To customize featurization step, provide a FeaturizationConfig object. Customized featurization currently supports blocking a set of transformers, updating column purpose, editing transformer parameters, and dropping columns.

https://docs.microsoft.com/en-us/python/api/azureml-train-automl-client

Select the algorithms to be searched

Large data

Automated ML supports a limited number of algorithms for training on large data that can successfully build models for big data on small virtual machines. Automated ML heuristics depend on properties such as data size, virtual machine memory size, experiment timeout and featurization settings to determine if these large data algorithms should be applied. Learn more about what models are supported in automated ML.

- For regression, Online Gradient Descent Regressor and Fast Linear Regressor
- For classification, Averaged Perceptron Classifier and Linear SVM Classifier; where the Linear SVM classifier has both large data and small data versions.
  If you want to override these heuristics, apply the following settings:

Task	**Setting**	**Notes**
Block data streaming algorithms	blocked_models in your AutoMLConfig object and list the model(s) you don't want to use.	Results in either run failure or long run time
Use data streaming algorithms	allowed_models in your AutoMLConfig object and list the model(s) you want to use.	
Use data streaming algorithms (studio UI experiments)	Block all models except the big data algorithms you want to use.	

Supported models
Automated machine learning tries different models and algorithms during the automation and tuning process. As a user, there is no need for you to specify the algorithm.

The three different task parameter values determine the list of algorithms, or models, to apply. Use the allowed_models or blocked_models parameters to further modify iterations with the available models to include or exclude. The following table summarizes the supported models by task type.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-train

Define a primary metric

Primary metric

The primary_metric parameter determines the metric to be used during model training for optimization. The available metrics you can select is determined by the task type you choose.

Choosing a primary metric for automated ML to optimize depends on many factors. We recommend your primary consideration be to choose a metric that best represents your business needs. Then consider if the metric is suitable for your dataset profile (data size, range, class distribution, etc.). The following sections summarize the recommended primary metrics based on task type and business scenario.

Metrics for classification scenarios

Threshold-dependent metrics, like accuracy, recall_score_weighted, norm_macro_recall, and precision_score_weighted may not optimize as well for datasets that are small, have very large class skew (class imbalance), or when the expected metric value is very close to 0.0 or 1.0. In those cases, AUC_weighted can be a better choice for the primary metric. After automated ML completes, you can choose the winning model based on the metric best suited to your business needs.

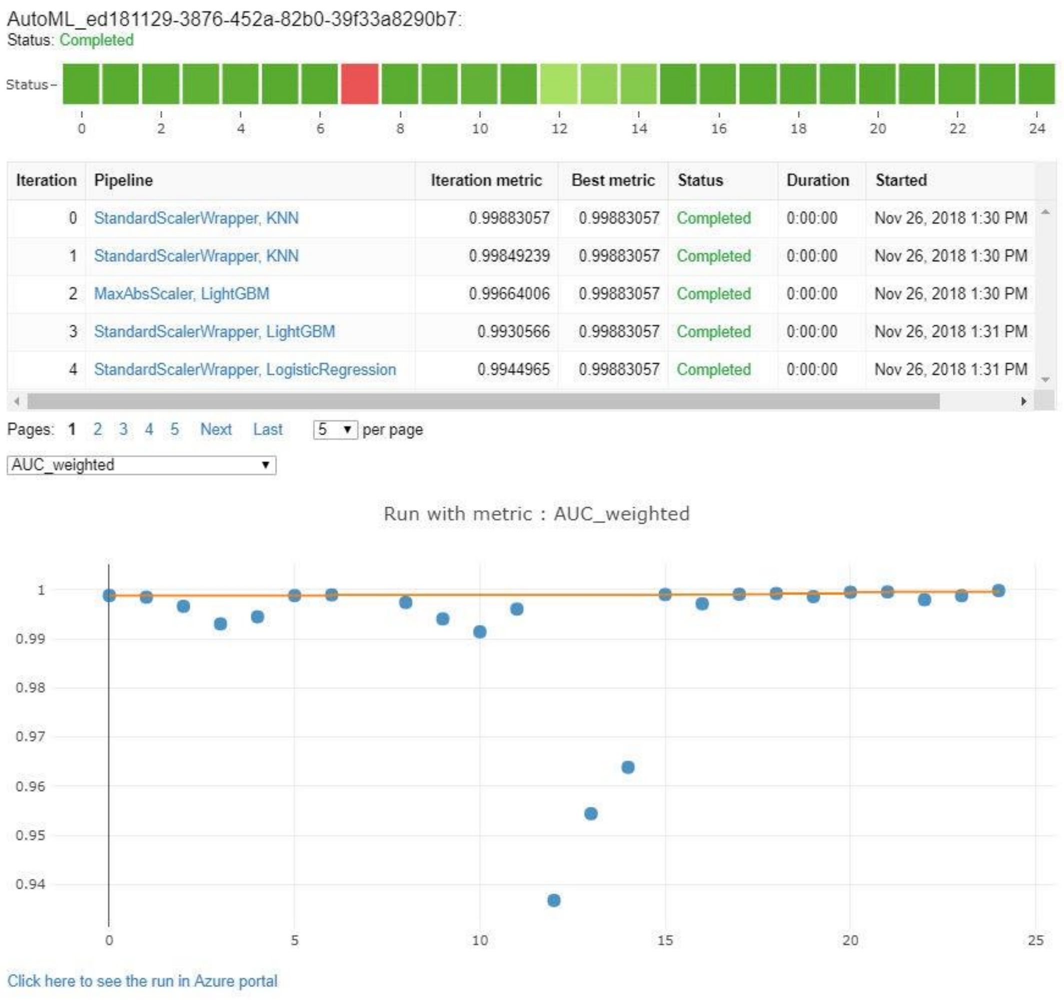
https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-train

Get data for an Automated ML run

For automated ML runs, to access the charts from a previous run, replace <<experiment_name>> with the appropriate experiment name:

Python

```python
from azureml.widgets import RunDetails
from azureml.core.run import Run

experiment = Experiment (workspace, <<experiment_name>>)
run_id = 'autoML_my_runID' #replace with run_ID
run = Run(experiment, run_id)
RunDetails(run).show()
```



AutoML_ed181129-3876-452a-82b0-39f33a8290b7:
Status: Completed

Iteration	Pipeline	Iteration metric	Best metric	Status	Duration	Started
0	StandardScalerWrapper, KNN	0.99883057	0.99883057	Completed	0:00:00	Nov 26, 2018 1:30 PM
1	StandardScalerWrapper, KNN	0.99849239	0.99883057	Completed	0:00:00	Nov 26, 2018 1:30 PM
2	MaxAbsScaler, LightGBM	0.99664006	0.99883057	Completed	0:00:00	Nov 26, 2018 1:30 PM
3	StandardScalerWrapper, LightGBM	0.9930566	0.99883057	Completed	0:00:00	Nov 26, 2018 1:31 PM
4	StandardScalerWrapper, LogisticRegression	0.9944965	0.99883057	Completed	0:00:00	Nov 26, 2018 1:31 PM

Pages: 1 2 3 4 5 Next Last 5 per page

AUC_weighted

Run with metric : AUC_weighted

Click here to see the run in Azure portal

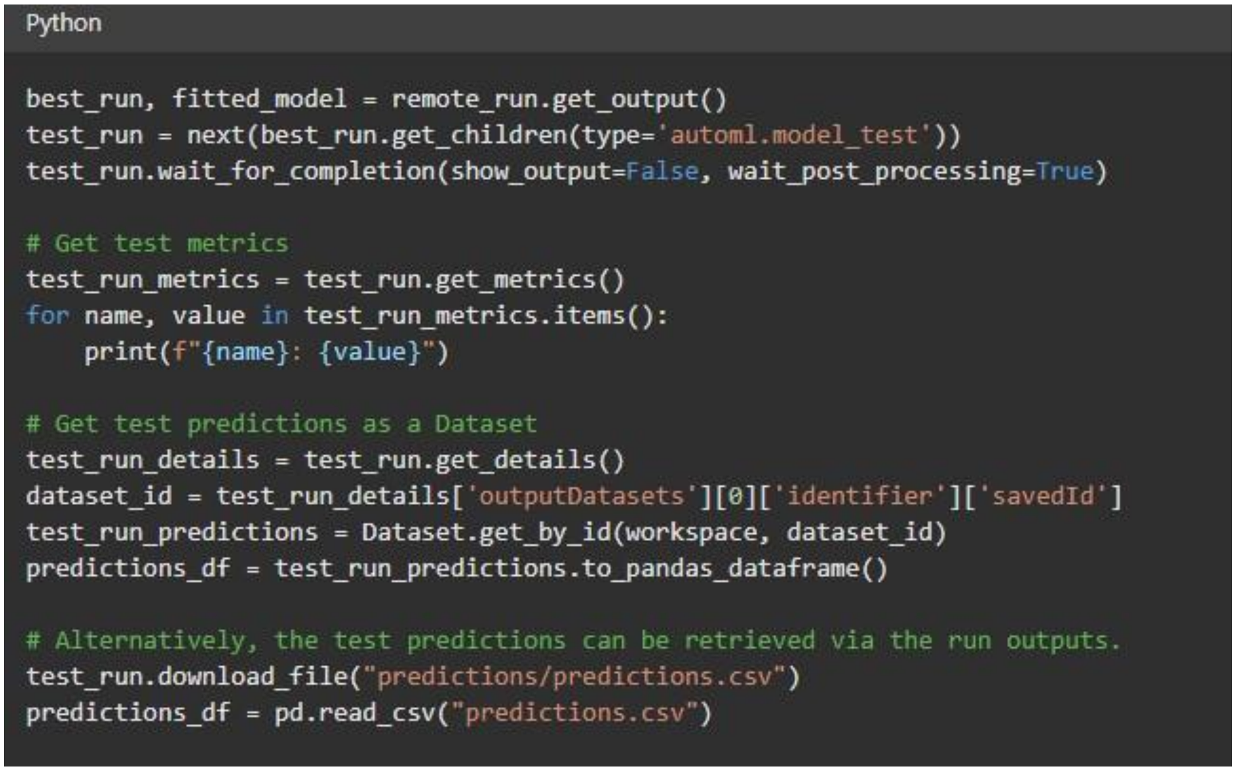
https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-train

Retrieve the best model

Passing the test_data or test_size parameters into the AutoMLConfig, automatically triggers a remote test run that uses the provided test data to evaluate the best model that automated ML recommends upon completion of the experiment. This remote test run is done at the end of the experiment, once the best model is determined. See how to pass test data into your AutoMLConfig.

Get test run results

You can get the predictions and metrics from the remote test run from the Azure Machine Learning studio or with the following code.



Python

```python
best_run, fitted_model = remote_run.get_output()
test_run = next(best_run.get_children(type='automl.model_test'))
test_run.wait_for_completion(show_output=False, wait_post_processing=True)

# Get test metrics
test_run_metrics = test_run.get_metrics()
for name, value in test_run_metrics.items():
    print(f"{name}: {value}")

# Get test predictions as a Dataset
test_run_details = test_run.get_details()
dataset_id = test_run_details['outputDatasets'][0]['identifier']['savedId']
test_run_predictions = Dataset.get_by_id(workspace, dataset_id)
predictions_df = test_run_predictions.to_pandas_dataframe()

# Alternatively, the test predictions can be retrieved via the run outputs.
test_run.download_file("predictions/predictions.csv")
predictions_df = pd.read_csv("predictions.csv")
```

The model test run generates the predictions.csv file that's stored in the default datastore created with the workspace. This datastore is visible to all users with the same subscription. Test runs are not recommended for scenarios if any of the information used for or created by the test run needs to remain private.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-train

Tune hyperparameters with Azure Machine Learning

Select a sampling method

The terms parameter and hyperparameter can be confusing. The model's parameters are what you set in the right pane of the component. Basically, this component performs a parameter sweep over the specified parameter settings. It learns an optimal set of hyperparameters, which might be different for each specific decision tree, dataset, or regression method. The process of finding the optimal configuration is sometimes called tuning.

The component supports the following method for finding the optimum settings for a model: integrated train and tune. In this method, you configure a set of parameters to use. You then let the component iterate over multiple combinations. The component measures accuracy until it finds a "best" model. With most learner components, you can choose which parameters should be changed during the training process, and which should remain fixed.

Depending on how long you want the tuning process to run, you might decide to exhaustively test all combinations. Or you might shorten the process by establishing a grid of parameter combinations and testing a randomized subset of the parameter grid.

Sampling the hyperparameter space

Specify the parameter sampling method to use over the hyperparameter space. Azure Machine Learning supports the following methods:

- Random sampling
- Grid sampling
- Bayesian sampling

Random sampling
Random sampling supports discrete and continuous hyperparameters. It supports early termination of low-performance runs. Some users do an initial search with random sampling and then refine the search space to improve results.

In random sampling, hyperparameter values are randomly selected from the defined search space.

Grid sampling
Grid sampling supports discrete hyperparameters. Use grid sampling if you can budget to exhaustively search over the search space. Supports early termination of low-performance runs.

Grid sampling does a simple grid search over all possible values. Grid sampling can only be used with choice hyperparameters.

Bayesian sampling
Bayesian sampling is based on the Bayesian optimization algorithm. It picks samples based on how previous samples did, so that new samples improve the primary metric.

Bayesian sampling is recommended if you have enough budget to explore the hyperparameter space. For best results, we recommend a maximum number of runs greater than or equal to 20 times the number of hyperparameters being tuned.

The number of concurrent runs has an impact on the effectiveness of the tuning process. A smaller number of concurrent runs may lead to better sampling convergence, since the smaller degree of parallelism increases the number of runs that benefit from previously completed runs.

Bayesian sampling only supports choice, uniform, and quniform distributions over the search space.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters

Define the search space

Tune hyperparameters by exploring the range of values defined for each hyperparameter. Hyperparameters can be discrete or continuous, and has a distribution of values described by a parameter expression.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters#define-the-search-space

Define the primary metric

Specify the primary metric you want hyperparameter tuning to optimize. Each training run is evaluated for the primary metric. The early termination policy uses the primary metric to identify low-performance runs.

Specify the following attributes for your primary metric:

- primary_metric_name: The name of the primary metric needs to exactly match the name of the metric logged by the training script primary_metric_goal: It can be either PrimaryMetricGoal.MAXIMIZE or PrimaryMetricGoal.MINIMIZE and determines whether the primary metric will be maximized or minimized when evaluating the runs.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters#specify-primary-metric-to-optimize

Define early termination options

Automatically end poorly performing runs with an early termination policy. Early termination improves computational efficiency.

You can configure the following parameters that control when a policy is applied:

- evaluation_interval: the frequency of applying the policy. Each time the training script logs the primary metric counts as one interval. An evaluation_interval of 1 will apply the policy every time the training script reports the primary metric. An evaluation_interval of 2 will apply the policy every other time. If not specified, evaluation_interval is set to 1 by default.
- delay_evaluation: delays the first policy evaluation for a specified number of intervals. This is an optional parameter that avoids premature termination of training runs by allowing all configurations to run for a minimum number of intervals. If specified, the policy applies every multiple of evaluation_interval that is greater than or equal to delay_evaluation.

Azure Machine Learning supports the following early termination policies:

Bandit policy
Bandit policy is based on slack factor/slack amount and evaluation interval. Bandit ends runs when the primary metric isn't within the specified slack factor/slack amount of the most successful run.

Median stopping policy
Median stopping is an early termination policy based on running averages of primary metrics reported by the runs. This policy computes running averages across all training runs and stops runs whose primary metric value is worse than the median of the averages.

Truncation selection policy
Truncation selection cancels a percentage of lowest performing runs at each evaluation interval. Runs are compared using the primary metric.

No termination policy
If no policy is specified, the hyperparameter tuning service will let all training runs execute to completion.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters

Find the model that has optimal hyperparameter values

Once all of the hyperparameter tuning runs have completed, identify the best performing configuration and hyperparameter values:

```python
best_run = hyperdrive_run.get_best_run_by_primary_metric()
best_run_metrics = best_run.get_metrics()
parameter_values = best_run.get_details()['runDefinition']['arguments']

print('Best Run Id: ', best_run.id)
print('\n Accuracy:', best_run_metrics['accuracy'])
print('\n learning rate:',parameter_values[3])
print('\n keep probability:',parameter_values[5])
print('\n batch size:',parameter_values[7])
```

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters

Deploy and operationalize machine learning solutions (35-40%)

Select compute for model deployment

Consider security for deployed services

You use HTTPS to restrict access to web services and secure the data that clients submit. HTTPS helps secure communications between a client and a web service by encrypting communications between the two. Encryption uses Transport Layer Security (TLS). TLS is sometimes still referred to as Secure Sockets Layer (SSL), which was the predecessor of TLS.

This is the general process to secure a web service:

1. Get a domain name.
2. Get a digital certificate.
3. Deploy or update the web service with TLS enabled.
4. Update your DNS to point to the web service.

Get a domain name
If you don't already own a domain name, purchase one from a *domain name registrar*. The process and price differ among registrars. The registrar provides tools to manage the domain name. You use these tools to map a fully qualified domain name (FQDN) (such as www.contoso.com) to the IP address that hosts your web service.

Get a TLS/SSL certificate
There are many ways to get an TLS/SSL certificate (digital certificate). The most common is to purchase one from a *certificate authority* (CA). Regardless of where you get the certificate, you need the following files:

- A **certificate**. The certificate must contain the full certificate chain, and it must be "PEM-encoded."
- A **key**. The key must also be PEM-encoded.

When you request a certificate, you must provide the FQDN of the address that you plan to use for the web service (for example, www.contoso.com). The address that's stamped into the certificate and the address that the clients use are compared to verify the identity of the web service. If those addresses don't match, the client gets an error message.

Enable TLS and deploy

For AKS deployment, you can enable TLS termination when you create or attach an AKS cluster in AML workspace. At AKS model deployment time, you can disable TLS termination with deployment configuration object, otherwise all AKS model deployment by default will have TLS termination enabled at AKS cluster create or attach time.

For ACI deployment, you can enable TLS termination at model deployment time with deployment configuration object.

Update your DNS
For either AKS deployment with custom certificate or ACI deployment, you must update your DNS record to point to the IP address of scoring endpoint.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-secure-web-service

Evaluate compute options for deployment

Workflow for deploying a model

The workflow is similar no matter where you deploy your model:

1. Register the model.
2. Prepare an entry script.
3. Prepare an inference configuration.
4. Deploy the model locally to ensure everything works.
5. Choose a compute target.
6. Deploy the model to the cloud.
7. Test the resulting web service.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where

Deploy a model as a service

Configure deployment settings

Define a deployment configuration

A deployment configuration specifies the amount of memory and cores your webservice needs in order to run. It also provides configuration details of the underlying webservice. For example, a deployment configuration lets you specify that your service needs 2 gigabytes of memory, 2 CPU cores, 1 GPU core, and that you want to enable autoscaling.

The options available for a deployment configuration differ depending on the compute target you choose. In a local deployment, all you can specify is which port your webservice will be served on.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where

Deploy a registered model

Deploy to cloud

Once you've confirmed your service works locally and chosen a remote compute target, you are ready to deploy to the cloud.

Change your deploy configuration to correspond to the compute target you've chosen, in this case Azure Container Instances:

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where

Deploy a model trained in Azure Databricks to an Azure Machine Learning endpoint

Set up a development environment with Azure Databricks and AutoML in Azure Machine Learning

Learn how to configure a development environment in Azure Machine Learning that uses Azure Databricks and automated ML.

Azure Databricks is ideal for running large-scale intensive machine learning workflows on the scalable Apache Spark platform in the Azure cloud. It provides a collaborative Notebook-based environment with a CPU or GPU-based compute cluster.

Azure Databricks with Azure Machine Learning and AutoML

Azure Databricks integrates with Azure Machine Learning and its AutoML capabilities.

You can use Azure Databricks:

- To train a model using Spark MLlib and deploy the model to ACI/AKS.
- With automated machine learning capabilities using an Azure ML SDK.
- As a compute target from an Azure Machine Learning pipeline.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-databricks-automl-environment

Consume a deployed service

Consume an Azure Machine Learning model deployed as a web service

Deploying an Azure Machine Learning model as a web service creates a REST API endpoint. You can send data to this endpoint and receive the prediction returned by the model. In this document, learn how to create clients for the web service by using C#, Go, Java, and Python.

You create a web service when you deploy a model to your local environment, Azure Container Instances, Azure Kubernetes Service, or field-programmable gate arrays (FPGA). You retrieve the URI used to access the web service by using the Azure Machine Learning SDK. If authentication is enabled, you can also use the SDK to get the authentication keys or tokens.

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.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-consume-web-service

Troubleshoot deployment container issues

Troubleshooting remote model deployment

Learn how to troubleshoot and solve, or work around, common errors you may encounter when deploying a model to Azure Container Instances (ACI) and Azure Kubernetes Service (AKS) using Azure Machine Learning.

If you are deploying a model to Azure Kubernetes Service (AKS), we advise you enable Azure Monitor for that cluster. This will help you understand overall cluster health and resource usage. You might also find the following resources useful:

Check for Resource Health events impacting your AKS cluster
Azure Kubernetes Service Diagnostics

If you are trying to deploy a model to an unhealthy or overloaded cluster, it is expected to experience issues. If you need help troubleshooting AKS cluster problems please contact AKS Support.

Debug locally

If you have problems when deploying a model to ACI or AKS, deploy it as a local web service. Using a local web service makes it easier to troubleshoot problems. To troubleshoot a deployment locally, see the local troubleshooting article.

Advanced debugging

You may need to interactively debug the Python code contained in your model deployment. For example, if the entry script is failing and the reason cannot be determined by additional logging. By using Visual Studio Code and the debugpy, you can attach to the code running inside the Docker container.

For more information, visit the interactive debugging in VS Code guide.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-troubleshoot-deployment

Manage models in Azure Machine Learning

Register a trained model

Register the model

A typical situation for a deployed machine learning service is that you need the following components:

- Resources representing the specific model that you want deployed (for example: a pytorch model file).
- Code that you will be running in the service, that executes the model on a given input.

Azure Machine Learnings allows you to separate the deployment into two separate components, so that you can keep the same code, but merely update the model. We define the mechanism by which you upload a model *separately* from your code as "registering the model".

When you register a model, we upload the model to the cloud (in your workspace's default storage account) and then mount it to the same compute where your webservice is running.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where

Monitor model usage

Monitor models with Azure Machine Learning

After a machine learning model has been deployed into production, it's important to understand how it is being used by capturing and viewing telemetry.

Exercise - Monitor a model

- 30 minutes

Now it's your chance to monitor a model that is deployed as an Azure Machine Learning real-time service.

In this exercise, you will:

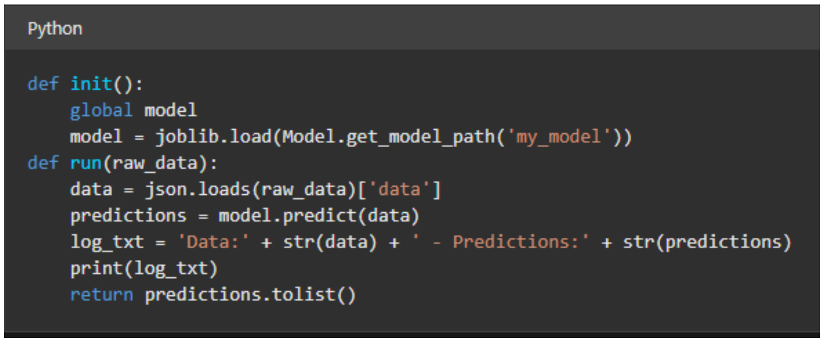
- Configure Application Insights for a deployed service.
- Capture and view model telemetry.

https://docs.microsoft.com/en-us/learn/modules/monitor-models-with-azure-machine-learning/4-monitor-models

Monitor data drift

Write log data

To capture telemetry data for Application insights, you can write any values to the standard output log in the scoring script for your service by using a print statement, as shown in the following example:



```python
def init():
    global model
    model = joblib.load(Model.get_model_path('my_model'))
def run(raw_data):
    data = json.loads(raw_data)['data']
    predictions = model.predict(data)
    log_txt = 'Data:' + str(data) + ' - Predictions:' + str(predictions)
    print(log_txt)
    return predictions.tolist()
```

Query logs in Application Insights

To analyze captured log data, you can use the Log Analytics query interface for Application Insights in the Azure portal. This interface supports a SQL-like query syntax that you can use to extract fields from logged data, including custom dimensions created by your Azure Machine Learning service.

For example, the following query returns the timestamp and customDimensions.Content fields from log traces that have a message field value of *STDOUT* (indicating the data is in the standard output log) and a customDimensions.["Service Name"] field value of *my-svc*:

SQL

```
traces
|where message == "STDOUT"
  and customDimensions.["Service Name"] = "my-svc"
| project  timestamp, customDimensions.Content
```

https://docs.microsoft.com/en-us/learn/modules/monitor-models-with-azure-machine-learning/3-capture-view-telemetry

Create an Azure Machine Learning pipeline for batch inferencing

Configure a ParallelRunStep

Create the data-preparation pipeline step

The first step in this pipeline will convert the compressed data files of fashion_ds into a dataset in your own workspace consisting of CSV files ready for use in training. Once registered with the workspace, your collaborators can access this data for their own analysis, training, and so on

Python

```python
datastore = workspace.get_default_datastore()
prepared_fashion_ds = OutputFileDatasetConfig(
    destination=(datastore, "outputdataset/{run-id}")
).register_on_complete(name="prepared_fashion_ds")
```

The above code specifies a dataset that is based on the output of a pipeline step. The underlying processed files will be put in the workspace's default datastore's blob storage at the path specified in destination. The dataset will be registered in the workspace with the name prepared_fashion_ds.

Create the pipeline step's source
The code that you've executed so far has create and controlled Azure resources. Now it's time to write code that does the first step in the domain.

If you're following along with the example in the AzureML Examples repo, the source file is already available as keras-mnist-fashion/prepare.py.

If you're working from scratch, create a subdirectory called kera-mnist-fashion/. Create a new file, add the following code to it, and name the file prepare.py.

Specify the pipeline step
Back in the Python environment you're using to specify the pipeline, run this code to create a PythonScriptStep for your preparation code:

The call to PythonScriptStep specifies that, when the pipeline step is run:

- All the files in the script_folder directory are uploaded to the compute_target
- Among those uploaded source files, the file prepare.py will be run
- The fashion_ds and prepared_fashion_ds datasets will be mounted on the compute_target and appear as directories
- The path to the fashion_ds files will be the first argument to prepare.py. In prepare.py, this argument is assigned to mounted_input_path
- The path to the prepared_fashion_ds will be the second argument to prepare.py. In prepare.py, this argument is assigned to mounted_output_path
- Because allow_reuse is True, it won't be rerun until its source files or inputs change
- This PythonScriptStep will be named prepare step

Modularity and reuse are key benefits of pipelines. Azure Machine Learning can automatically determine source code or Dataset changes. The output of a step that isn't affected will be reused without rerunning the steps again if allow_reuse is True. If a step relies on a data source external to Azure Machine Learning that may change (for instance, a URL that contains sales data), set allow_reuse to False and the pipeline step will run every time the pipeline is run.

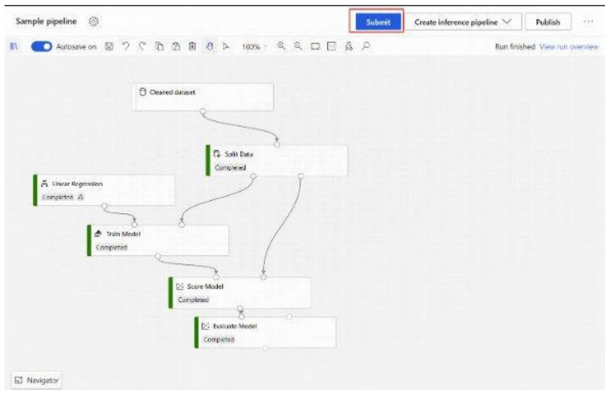
https://docs.microsoft.com/en-us/azure/machine-learning/tutorial-pipeline-python-sdk

Configure compute for a batch inferencing pipeline

Create a batch inference pipeline

Your training pipeline must be run at least once to be able to create an inferencing pipeline.

1. Go to the **Designer** tab in your workspace.
2. Select the training pipeline that trains the model you want to use to make prediction.
3. **Submit** the pipeline.



Now that the training pipeline has been run, you can create a batch inference pipeline.

1. Next to **Submit**, select the new dropdown **Create inference pipeline**.
2. Select **Batch inference pipeline**.

The result is a default batch inference pipeline.

Add a pipeline parameter
To create predictions on new data, you can either manually connect a different dataset in this pipeline draft view or create a parameter for your dataset. Parameters let you change the behavior of the batch inferencing process at runtime.

In this section, you create a dataset parameter to specify a different dataset to make predictions on.

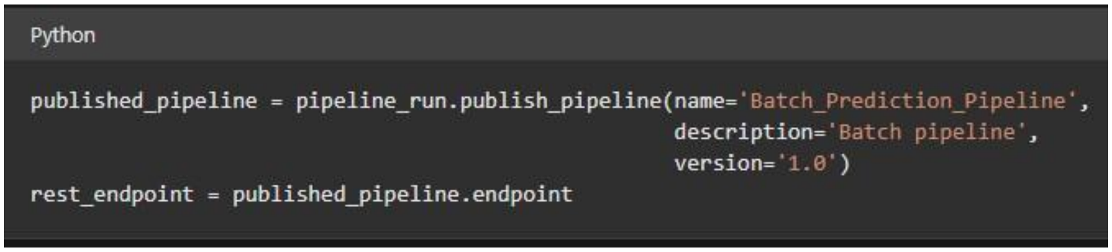
1. Select the dataset component.
2. A pane will appear to the right of the canvas. At the bottom of the pane, select **Set as pipeline parameter**.
   Enter a name for the parameter, or accept the default value.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-run-batch-predictions-designer

Publish a batch inferencing pipeline

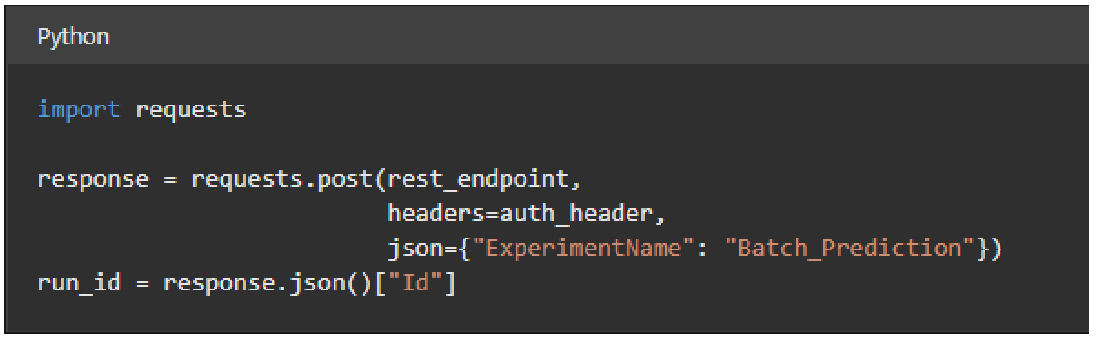
Publishing a batch inference pipeline

You can publish a batch inferencing pipeline as a REST service, as shown in the following example code:


Python

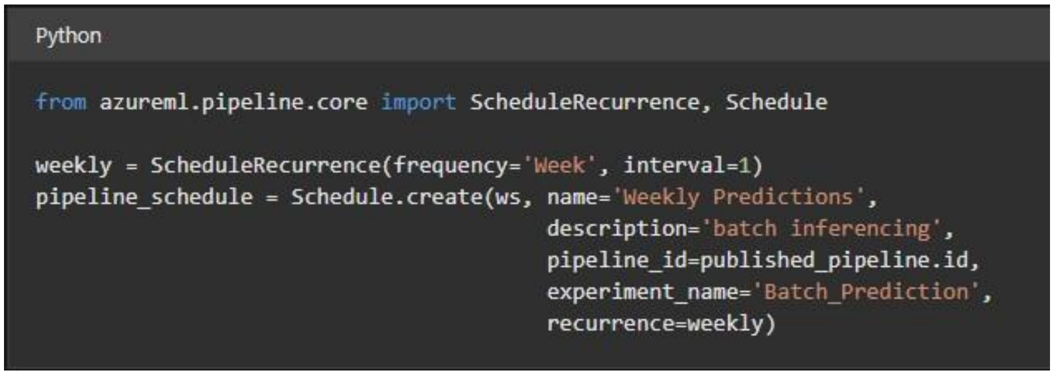
```python
published_pipeline = pipeline_run.publish_pipeline(name='Batch_Prediction_Pipeline',
                                                   description='Batch pipeline',
                                                   version='1.0')
rest_endpoint = published_pipeline.endpoint
```

Once published, you can use the service endpoint to initiate a batch inferencing job, as shown in the following example code:


Python

```python
import requests

response = requests.post(rest_endpoint,
                         headers=auth_header,
                         json={"ExperimentName": "Batch_Prediction"})
run_id = response.json()["Id"]
```

You can also schedule the published pipeline to have it run automatically, as shown in the following example code:


Python

```python
from azureml.pipeline.core import ScheduleRecurrence, Schedule

weekly = ScheduleRecurrence(frequency='Week', interval=1)
pipeline_schedule = Schedule.create(ws, name='Weekly Predictions',
                                        description='batch inferencing',
                                        pipeline_id=published_pipeline.id,
                                        experiment_name='Batch_Prediction',
                                        recurrence=weekly)
```

https://docs.microsoft.com/en-us/learn/modules/deploy-batch-inference-pipelines-with-azure-machine-learning/3-publish-batch-pipeline

Run a batch inferencing pipeline and obtain outputs

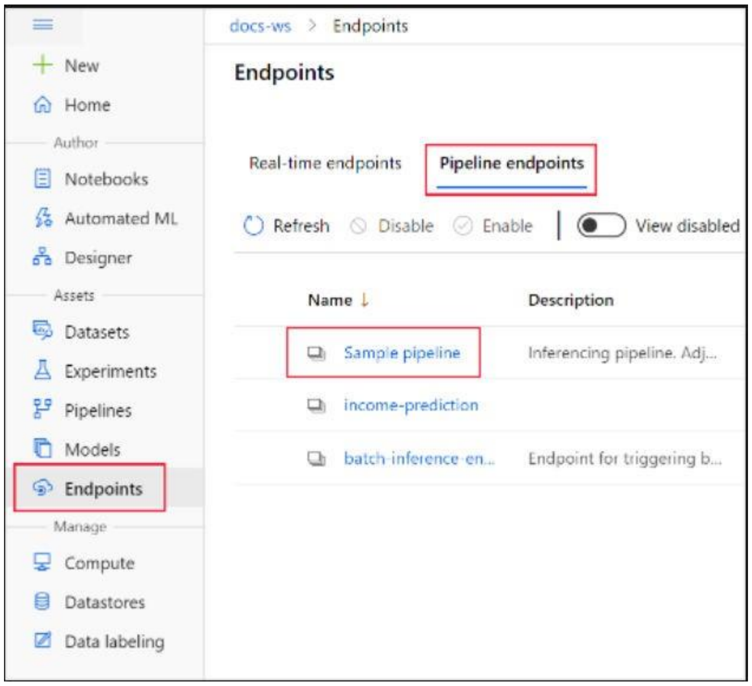
Publish your batch inference pipeline

Now you're ready to deploy the inference pipeline. This will deploy the pipeline and make it available for others to use.

1. Select the **Publish** button.
2. In the dialog that appears, expand the drop-down for **PipelineEndpoint**, and select **New PipelineEndpoint**.
3. Provide an endpoint name and optional description.
   Near the bottom of the dialog, you can see the parameter you configured with a default value of the dataset ID used during training.
4. Select **Publish**.

Submit a pipeline run
In this section, you will set up a manual pipeline run and alter the pipeline parameter to score new data.

1. After the deployment is complete, go to the **Endpoints** section.
2. Select **Pipeline endpoints**.
3. Select the name of the endpoint you created.



Name ↓	Description
Sample pipeline	Inferencing pipeline. Adj...
income-prediction	
batch-inference-en...	Endpoint for triggering b...

1. Select **Published pipelines**.

   This screen shows all published pipelines published under this endpoint.
2. Select the pipeline you published.

   The pipeline details page shows you a detailed run history and connection string information for your pipeline.
3. Select **Submit** to create a manual run of the pipeline.
4. Change the parameter to use a different dataset.
5. Select **Submit** to run the pipeline.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-run-batch-predictions-designer?WT.mc_id=AI-MVP-5004069#publish-your-batch-inference-pipeline

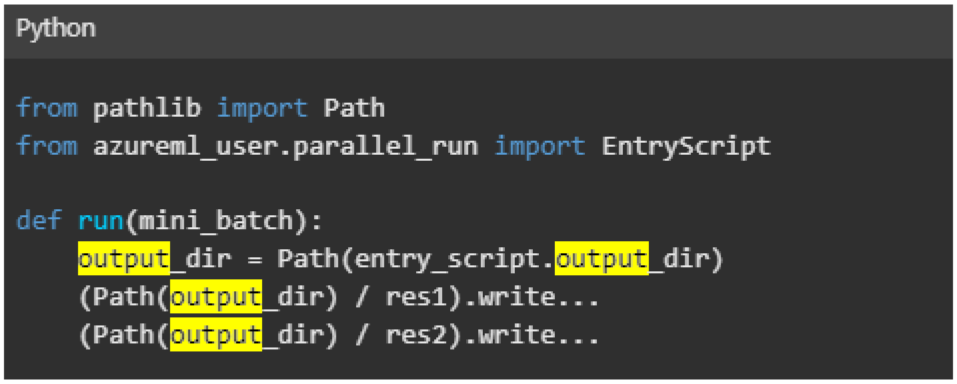
Obtain outputs from a ParallelRunStep

Create the ParallelRunStep by using the script, environment configuration, and parameters. Specify the compute target that you already attached to your workspace as the target of execution for your inference script.

output: An OutputFileDatasetConfig object that represents the directory path at which the output data will be stored.

You can get the output directory from the EntryScript class and write to it. To view the written files, in the step Run view in the Azure Machine Learning portal, select the Outputs + logs tab. Select the Data outputs link, and then complete the steps that are described in the dialog.

Use EntryScript in your entry script like in this example:


Python

```
from pathlib import Path
from azureml_user.parallel_run import EntryScript

def run(mini_batch):
    output_dir = Path(entry_script.output_dir)
    (Path(output_dir) / res1).write...
    (Path(output_dir) / res2).write...
```

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-debug-parallel-run-step

Publish an Azure Machine Learning designer pipeline as a web service

Create a target compute resource

With Azure Machine Learning, you can train your model on various resources or environments, collectively referred to as compute targets. A compute target can be a local machine or a cloud resource, such as an Azure Machine Learning Compute, Azure HDInsight, or a remote virtual machine. You also use compute targets for model deployment as described in "Where and how to deploy your models".

Local computer
When you use your local computer for training, there is no need to create a compute target. Just submit the training run from your local machine.

When you use your local computer for inference, you must have Docker installed. To perform the deployment, use LocalWebservice.deploy_configuration() to define the port that the web service will use. Then use the normal deployment process as described in Deploy models with Azure Machine Learning.

Remote virtual machines
Azure Machine Learning also supports attaching an Azure Virtual Machine. The VM must be an Azure Data Science Virtual Machine (DSVM). The VM offers a curated choice of tools and frameworks for full-lifecycle machine learning development. For more information on how to use the DSVM with Azure Machine Learning, see Configure a development environment.

1. Create: Azure Machine Learning cannot create a remote VM for you. Instead, you must create the VM and then attach it to your Azure Machine Learning workspace. For information on creating a DSVM, see Provision the Data Science Virtual Machine for Linux (Ubuntu).

2. Attach: To attach an existing virtual machine as a compute target, you must provide the resource ID, user name, and password for the virtual machine. The resource ID of the VM can be constructed using the subscription ID, resource group name, and VM name using the following string format: /subscriptions/<subscription_id>/resourceGroups/<resource_group>/providers/Microsoft.Compute/virtualMachines/<vm_name>

3. Configure: Create a run configuration for the DSVM compute target. Docker and conda are used to create and configure the training environment on the DSVM.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-attach-compute-targets

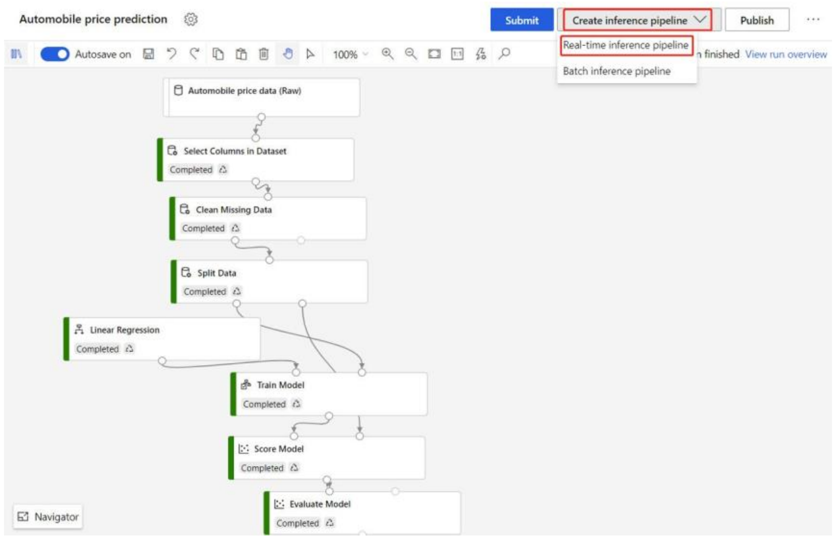
Configure an inference pipeline

Create a real-time inference pipeline

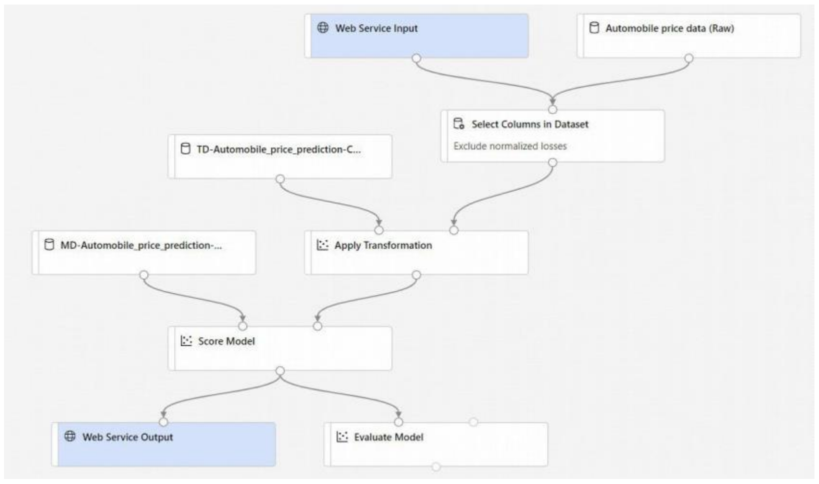
To deploy your pipeline, you must first convert the training pipeline into a real-time inference pipeline. This process removes training components and adds web service inputs and outputs to handle requests.

Create a real-time inference pipeline

1. Above the pipeline canvas, select **Create inference pipeline** > **Real-time inference pipeline**.



Your pipeline should now look like this:


Web Service Input
Automobile price data (Raw)
Select Columns in Dataset
Exclude normalized losses
TD-Automobile_price_prediction-C...
MD-Automobile_price_prediction-...
Apply Transformation
Score Model
Web Service Output
Evaluate Model

When you select **Create inference pipeline**, several things happen:

- The trained model is stored as a **Dataset** component in the component palette. You can find it under **My Datasets**.
- Training components like **Train Model** and **Split Data** are removed.
- The saved trained model is added back into the pipeline.
- **Web Service Input** and **Web Service Output** components are added. These components show where user data enters the pipeline and where data is returned.

2. Select **Submit**, and use the same compute target and experiment that you used in part one.
If this is the first run, it may take up to 20 minutes for your pipeline to finish running. The default compute settings have a minimum node size of 0, which means that the designer must allocate resources after being idle. Repeated pipeline runs will take less time since the compute resources are already allocated. Additionally, the designer uses cached results for each component to further improve efficiency.

3. Select **Deploy**.

https://docs.microsoft.com/en-us/azure/machine-learning/tutorial-designer-automobile-price-deploy

Consume a deployed endpoint

Deploying an Azure Machine Learning model as a web service creates a REST API endpoint. You can send data to this endpoint and receive the prediction returned by the model. In this document, learn how to create clients for the web service by using C#, Go, Java, and Python.

You create a web service when you deploy a model to your local environment, Azure Container Instances, Azure Kubernetes Service, or field-programmable gate arrays (FPGA). You retrieve the URI used to access the web service by using the Azure Machine Learning SDK. If authentication is enabled, you can also use the SDK to get the authentication keys or tokens.

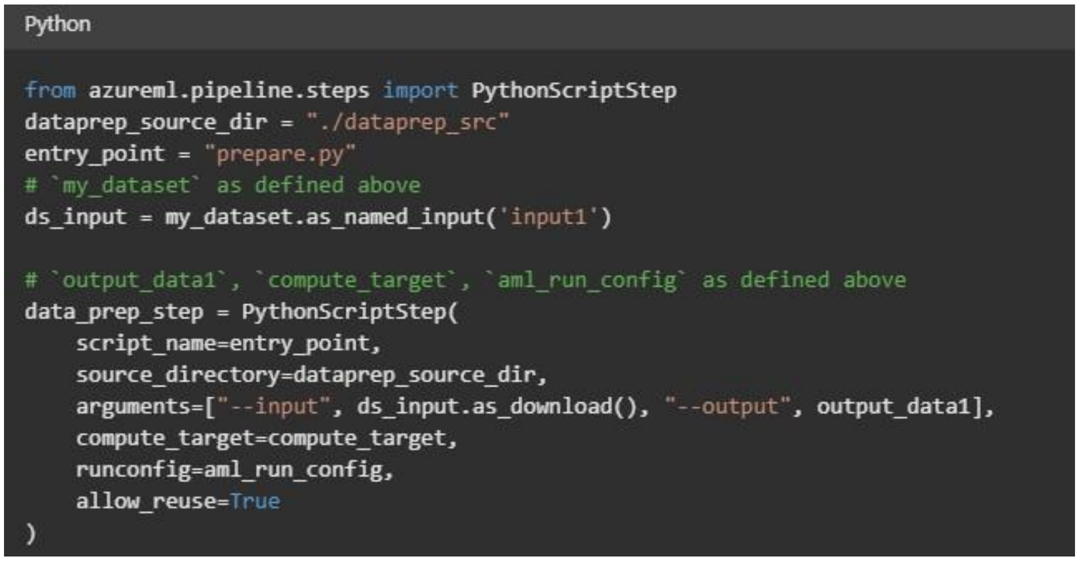
The general workflow for creating a client that uses a machine learning web service is:
- Use the SDK to get the connection information.
- Determine the type of request data used by the model.
- Create an application that calls the web service.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-consume-web-service

Implement pipelines by using the Azure Machine Learning SDK

Create a pipeline

Once you have the compute resource and environment created, you're ready to define your pipeline's steps. There are many built-in steps available via the Azure Machine Learning SDK, as you can see on the reference documentation for the azureml.pipeline.steps package. The most flexible class is PythonScriptStep, which runs a Python script.



Python

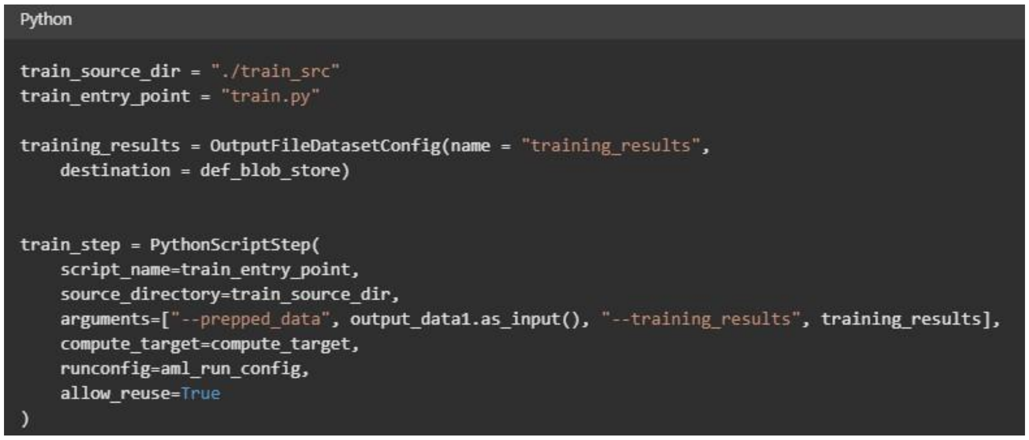
```python
from azureml.pipeline.steps import PythonScriptStep
dataprep_source_dir = "./dataprep_src"
entry_point = "prepare.py"
# `my_dataset` as defined above
ds_input = my_dataset.as_named_input('input1')

# `output_data1`, `compute_target`, `aml_run_config` as defined above
data_prep_step = PythonScriptStep(
    script_name=entry_point,
    source_directory=dataprep_source_dir,
    arguments=["--input", ds_input.as_download(), "--output", output_data1],
    compute_target=compute_target,
    runconfig=aml_run_config,
    allow_reuse=True
)
```

The above code shows a typical initial pipeline step. Your data preparation code is in a subdirectory (in this example, "prepare.py" in the directory "./dataprep.src"). As part of the pipeline creation process, this directory is zipped and uploaded to the compute_target and the step runs the script specified as the value for script_name.

The arguments values specify the inputs and outputs of the step. In the example above, the baseline data is the my_dataset dataset. The corresponding data will be downloaded to the compute resource since the code specifies it as as_download(). The script prepare.py does whatever data-transformation tasks are appropriate to the task at hand and outputs the data to output_data1, of type OutputFileDatasetConfig. For more information, see Moving data into and between ML pipeline steps (Python). The step will run on the machine defined by compute_target, using the configuration aml_run_config.

Reuse of previous results (allow_reuse) is key when using pipelines in a collaborative environment since eliminating unnecessary reruns offers agility. Reuse is the default behavior when the script_name, inputs, and the parameters of a step remain the same. When reuse is allowed, results from the previous run are immediately sent to the next step. If allow_reuse is set to False, a new run will always be generated for this step during pipeline execution.

It's possible to create a pipeline with a single step, but almost always you'll choose to split your overall process into several steps. For instance, you might have steps for data preparation, training, model comparison, and deployment. For instance, one might imagine that after the data_prep_step specified above, the next step might be training:



Python

```python
train_source_dir = "./train_src"
train_entry_point = "train.py"

training_results = OutputFileDatasetConfig(name = "training_results",
    destination = def_blob_store)


train_step = PythonScriptStep(
    script_name=train_entry_point,
    source_directory=train_source_dir,
    arguments=["--prepped_data", output_data1.as_input(), "--training_results", training_results],
    compute_target=compute_target,
    runconfig=aml_run_config,
    allow_reuse=True
)
```

The above code is similar to the code in the data preparation step. The training code is in a directory separate from that of the data preparation code.

The OutputFileDatasetConfig output of the data preparation step, output_data1 is used as the input to the training step. A new OutputFileDatasetConfig object, training_results is created to hold the results for a later comparison or deployment step.

For other code examples, see how to build a two step ML pipeline and how to write data back to datastores upon run completion.

After you define your steps, you build the pipeline by using some or all of those steps.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-machine-learning-pipelines

Pass data between steps in a pipeline

To pass the dataset's path to your script, use the Dataset object's as_named_input() method. You can either pass the resulting DatasetConsumptionConfig object to your script as an argument or, by using the inputs argument to your pipeline script, you can retrieve the dataset using Run.get_context().input_datasets[].

Once you've created a named input, you can choose its access mode: as_mount() or as_download

(). If your script processes all the files in your dataset and the disk on your compute resource is large enough for the dataset, the download access mode is the better choice. The download access mode will avoid the overhead of streaming the data at runtime. If your script accesses a subset of the dataset or it's too large for your compute, use the mount access mode. For more information, read Mount vs. Download

To pass a dataset to your pipeline step:

1. Use TabularDataset.as_named_input() or FileDataset.as_named_input() (no 's' at end) to create a DatasetConsumptionConfig object

2. Use as_mount() or as_download() to set the access mode

Pass the datasets to your pipeline steps using either the arguments or the inputs argument

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-move-data-in-out-of-pipelines

Run a pipeline

When you first run a pipeline, Azure Machine Learning:

•Downloads the project snapshot to the compute target from the Blob storage associated with the workspace.

•Builds a Docker image corresponding to each step in the pipeline.

•Downloads the Docker image for each step to the compute target from the container registry.

•Configures access to Dataset and OutputFileDatasetConfig objects. For as_mount() access mode, FUSE is used to provide virtual access. If mount isn't supported or if the user specified access as as_upload(), the data is instead copied to the compute target.

•Runs the step in the compute target specified in the step definition.

•Creates artifacts, such as logs, stdout and stderr, metrics, and output specified by the step. These artifacts are then uploaded and kept in the user's default datastore.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-machine-learning-pipelines

Monitor pipeline runs

The OpenCensus Python library can be used to route logs to Application Insights from your scripts. Aggregating logs from pipeline runs in one place allows you to build queries and diagnose issues. Using Application Insights will allow you to track logs over time and compare pipeline logs across runs.

Having your logs in once place will provide a history of exceptions and error messages. Since Application Insights integrates with Azure Alerts, you can also create alerts based on Application Insights queries.

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-log-pipelines-application-insights

Apply ML Ops practices

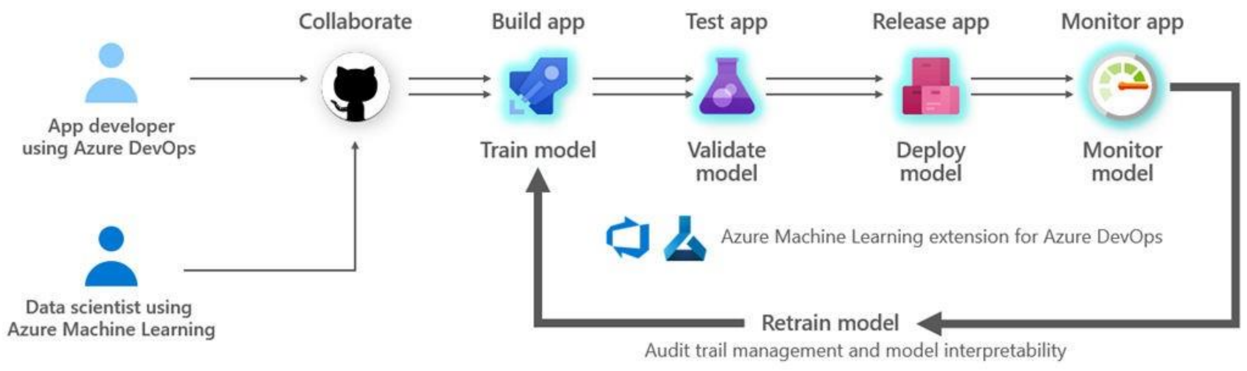
Trigger an Azure Machine Learning pipeline from Azure DevOps

Azure Machine Learning integrated with Azure DevOps for you to be able to create MLOps pipelines inside the DevOps environment. Azure DevOps has an extension for Azure Machine Learning, which enables it to listen to Azure Machine Learning's model registry in addition to the code repository maintained in GitHub for the python notebooks and scripts. This enables to trigger Azure Pipelines based on new code commits into the code repository or new models published into the model repository. This is extremely powerful, as data science teams can configure stages for build and release pipelines within Azure DevOps for the machine learning models and completely automate the process.

What's more, since Azure DevOps is also the environment to manage app lifecycles it now enables data science teams and app dev teams to collaborate seamlessly and trigger new version of the apps whenever certain conditions are met for the MLOps cycle, as they are the ones often leveraging the new versions of the ML models, infusing them into apps or updating inference call URLs, when desired.

This may sound simple and the most logical way of doing it, but nobody has been able to bring MLOps to life with such close-knit integration into the whole process. Azure Machine Learning does an amazing job of it enabling data science teams to become immensely productive.

Please see the diagrammatic representation below for MLOps with Azure Machine Learning.


Collaborate
Build app
Test app
Release app
Monitor app
App developer using Azure DevOps
Train model
Validate model
Deploy model
Monitor model
Azure Machine Learning extension for Azure DevOps
Data scientist using Azure Machine Learning
Retrain model
Audit trail management and model interpretability

Visual: MLOps with Azure Machine Learning

Automate model retraining based on new data additions or data changes

Customers working with Azure Machine Learning models have been leveraging the built in AzureMLBatchExecution activity with Azure Data Factory pipelines to operationalize the ML models in production and score new data against the pre-trained models at scale. But as trends and variables that influence the model's parameters change over time, ideally this pipeline should also support recurring automated retraining and updates to the model with latest training data. Now Azure Data Factory allows you to do just that with the newly released AzureMLUpdateResource activity.

With Azure ML you typically first setup your scoring and training experiments, then two separate web service endpoints for each experiment. Next, you can use the AzureMLBatchExecution activity with Data Factory to do both scoring of incoming data against the latest model hosted by the scoring web service and scheduled retraining with latest training data. The scoring web service endpoint also exposes an Update Resource method that can be used to update the model used by the scoring web service. This is where the new AzureMLUpdateResource activity comes into picture. You can use this activity now to take the model generated by the training activity and provide it to the scoring web service to update the model for scoring, on a schedule, all automated with your existing data factory pipeline.

https://azure.microsoft.com/en-us/blog/retraining-and-updating-azure-machine-learning-models-with-azure-data-factory/?WT.mc_id=AI-MVP-5004069

Refactor notebooks into scripts

The Jupyter code needs to be refactored into functions. Refactoring code into functions makes unit testing easier and makes the code more maintainable. In this section, you'll refactor:

The Diabetes Ridge Regression Training notebook(experimentation/Diabetes Ridge Regression Training.ipynb)

The Diabetes Ridge Regression Scoring notebook(experimentation/Diabetes Ridge Regression Scoring.ipynb)

https://docs.microsoft.com/en-us/azure/machine-learning/tutorial-convert-ml-experiment-to-production

Implement source control for scripts

Git is a popular version control system that allows you to share and collaborate on your projects.

Azure Machine Learning fully supports Git repositories for tracking work - you can clone repositories directly onto your shared workspace file system, use Git on your local workstation, or use Git from a CI/CD pipeline.

When submitting a job to Azure Machine Learning, if source files are stored in a local git repository then information about the repo is tracked as part of the training process.

Since Azure Machine Learning tracks information from a local git repo, it isn't tied to any specific central repository. Your repository can be cloned from GitHub, GitLab, Bitbucket, Azure DevOps, or any other git-compatible service.

https://docs.microsoft.com/en-us/azure/machine-learning/concept-train-model-git-integration

Implement responsible machine learning (5-10%)

Use model explainers to interpret models

Select a model interpreter

azureml-interpret uses the interpretability techniques developed in Interpret-Community, an open source Python package for training interpretable models and helping to explain blackbox AI systems. Interpret-Community serves as the host for this SDK's supported explainers, and currently supports the following interpretability techniques:

Interpretability Technique	Description	Type
SHAP Tree Explainer	SHAP's tree explainer, which focuses on polynomial time fast SHAP value estimation algorithm specific to **trees and ensembles of trees**.	Model-specific
SHAP Deep Explainer	Based on the explanation from SHAP, Deep Explainer "is a high-speed approximation algorithm for SHAP values in deep learning models that builds on a connection with DeepLIFT described in the SHAP NIPS paper. **TensorFlow** models and **Keras** models using the TensorFlow backend are supported (there is also preliminary support for PyTorch)".	Model-specific
SHAP Linear Explainer	SHAP's Linear explainer computes SHAP values for a **linear model**, optionally accounting for inter-feature correlations.	Model-specific
SHAP Kernel Explainer	SHAP's Kernel explainer uses a specially weighted local linear regression to estimate SHAP values for **any model**.	Model-agnostic
Mimic Explainer (Global Surrogate)	Mimic explainer is based on the idea of training global surrogate models to mimic blackbox models. A global surrogate model is an intrinsically interpretable model that is trained to approximate the predictions of **any black box model** as accurately as possible. Data scientists can interpret the surrogate model to draw conclusions about the black box model. You can use one of the following interpretable models as your surrogate model: LightGBM (LGBMExplainableModel), Linear Regression (LinearExplainableModel), Stochastic Gradient Descent explainable model (SGDExplainableModel),	Model-agnostic

Permutation Feature Importance Explainer (PFI)	Permutation Feature Importance is a technique used to explain classification and regression models that is inspired by Breiman's Random Forests paper (see section 10). At a high level, the way it works is by randomly shuffling data one feature at a time for the entire dataset and calculating how much the performance metric of interest changes. The larger the change, the more important that feature is. PFI can explain the overall behavior of **any underlying model** but does not explain individual predictions.	Model-agnostic

https://docs.microsoft.com/en-us/azure/machine-learning/how-to-machine-learning-interpretability

Generate feature importance data

How to use Permutation Feature Importance

Generating a set of feature scores requires that you have an already trained model, as well as a test dataset.

1. Add the Permutation Feature Importance component to your pipeline. You can find this component in the **Feature Selection** category.

2. Connect a trained model to the left input. The model must be a regression model or a classification model.

3. On the right input, connect a dataset. Preferably, choose one that's different from the dataset that you used for training the model. This dataset is used for scoring based on the trained model. It's also used for evaluating the model after feature values have changed.

4. For **Random seed**, enter a value to use as a seed for randomization. If you specify 0 (the default), a number is generated based on the system clock.
A seed value is optional, but you should provide a value if you want reproducibility across runs of the same pipeline.

5. For **Metric for measuring performance**, select a single metric to use when you're computing model quality after permutation.
Azure Machine Learning designer supports the following metrics, depending on whether you're evaluating a classification or regression model:

- **Classification**
- Accuracy, Precision, Recall
- **Regression**

Precision, Recall, Mean Absolute Error, Root Mean Squared Error, Relative Absolute Error, Relative Squared Error, Coefficient of Determination

6. Submit the pipeline.

7. The component outputs a list of feature columns and the scores associated with them. The list is ranked in descending order of the scores.

https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/permutation-feature-importance

Describe fairness considerations for models

Evaluate model fairness based on prediction disparity

Fairness assessment and mitigation with Fairlearn
Fairlearn is an open-source Python package that allows machine learning systems developers to assess their systems' fairness and mitigate unfairness.

The Fairlearn open-source package has two components:

- Assessment Dashboard: A Jupyter notebook widget for assessing how a model's predictions affect different groups. It also enables comparing multiple models by using fairness and performance metrics.
- Mitigation Algorithms: A set of algorithms to mitigate unfairness in binary classification and regression.

Together, these components enable data scientists and business leaders to navigate any trade -offs between fairness and performance, and to select the mitigation strategy that best fits their needs.

Assess fairness in machine learning models
In the Fairlearn open-source package, fairness is conceptualized through an approach known as group fairness, which asks: Which groups of individuals are at risk for experiencing harms? The relevant groups, also known as subpopulations, are defined through sensitive features or sensitive attributes. Sensitive features are passed to an estimator in the Fairlearn open-source package as a vector or a matrix called sensitive_features. The term suggests that the system designer should be sensitive to these features when assessing group fairness.
Something to be mindful of is whether these features contain privacy implications due to private data. But the word "sensitive" doesn't imply that these features shouldn't be used to make predictions.

During assessment phase, fairness is quantified through disparity metrics. **Disparity metrics** can evaluate and compare model's behavior across different groups either as ratios or as differences. The Fairlearn open-source package supports two classes of disparity metrics:

- Disparity in model performance: These sets of metrics calculate the disparity (difference) in the values of the selected performance metric across different subgroups. Some examples include:

◊ disparity in accuracy rate
◊ disparity in error rate
◊ disparity in precision
◊ disparity in recall
◊ disparity in MAE
◊ many others

- Disparity in selection rate: This metric contains the difference in selection rate among different subgroups. An example of this is disparity in loan approval rate. Selection rate means the fraction of datapoints in each class classified as 1 (in binary classification) or distribution of prediction values (in regression).

https://docs.microsoft.com/en-us/azure/machine-learning/concept-fairness-ml

Mitigate model unfairness

Parity constraints
The Fairlearn open-source package includes a variety of unfairness mitigation algorithms. These algorithms support a set of constraints on the predictor's behavior called **parity constraints** or criteria. Parity constraints require some aspects of the predictor behavior to be comparable across the groups that sensitive features define (for example, different races). The mitigation algorithms in the Fairlearn open-source package use such parity constraints to mitigate the observed fairness issues.

Mitigation algorithms
The Fairlearn open-source package provides postprocessing and reduction unfairness mitigation algorithms:

- Reduction: These algorithms take a standard black-box machine learning estimator (for example, a LightGBM model) and generate a set of retrained models using a sequence of re-weighted training datasets. For example, applicants of a certain gender might be up-weighted or down-weighted to retrain models and reduce disparities across different gender groups. Users can then pick a model that provides the best trade-off between accuracy (or other performance metric) and disparity, which generally would need to be based on business rules and cost calculations.

- Post-processing: These algorithms take an existing classifier and the sensitive feature as input. Then, they derive a transformation of the classifier's prediction to enforce the specified fairness constraints. The biggest advantage of threshold optimization is its simplicity and flexibility as it doesn't need to retrain the model.

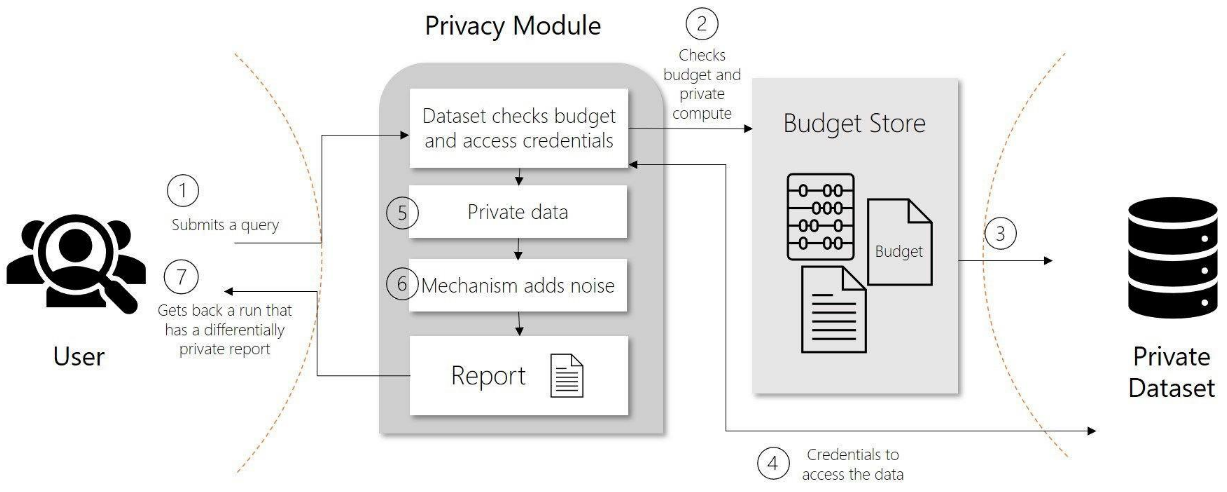
https://docs.microsoft.com/en-us/azure/machine-learning/concept-fairness-ml

Describe privacy considerations for data

Describe principles of differential privacy

As the amount of data that an organization collects and uses for analyses increases, so do concerns of privacy and security. Analyses require data. Typically, the more data used to train machine learning models, the more accurate they are. When personal information is used for these analyses, it's especially important that the data remains private throughout its use.

Differential privacy is a set of systems and practices that help keep the data of individuals safe and private. In machine learning solutions, differential privacy may be required for regulatory compliance.


Privacy Module
2
Checks budget and private compute
Dataset checks budget and access credentials
Budget Store
1
Submits a query
5
Private data
Budget
3
6
Mechanism adds noise
7
Gets back a run that has a differentially private report
User
Report
Private Dataset
4
Credentials to access the data

In traditional scenarios, raw data is stored in files and databases. When users analyze data, they typically use the raw data. This is a concern because it might infringe on an individual's privacy. Differential privacy tries to deal with this problem by adding "noise" or randomness to the data so that users can't identify any individual data points. At the least, such a system provides plausible deniability. Therefore, the privacy of individuals is preserved with limited impact on the accuracy of the data.

In differentially private systems, data is shared through requests called queries. When a user submits a query for data, operations known as privacy mechanisms add noise to the requested data. Privacy mechanisms return an approximation of the data instead of the raw data. This privacy-preserving result appears in a report. Reports consist of two parts, the actual data computed and a description of how the data was created.

https://docs.microsoft.com/en-us/azure/machine-learning/concept-differential-privacy|

Specify acceptable levels of noise in data and the effects on privacy

SmartNoise is an open-source project that contains components for building machine learning solutions with differential privacy. SmartNoise is made up of the following top-level components:

- SmartNoise Core library
- SmartNoise SDK library

SmartNoise Core
The core library includes the following privacy mechanisms for implementing a differentially private system:

Component	**Description**
Analysis	A graph description of arbitrary computations.
Validator	A Rust library that contains a set of tools for checking and deriving the necessary conditions for an analysis to be differentially private.
Runtime	The medium to execute the analysis. The reference runtime is written in Rust but runtimes can be written using any computation framework such as SQL and Spark depending on your data needs.
Bindings	Language bindings and helper libraries to build analyses. Currently SmartNoise provides Python bindings.

https://docs.microsoft.com/en-us/azure/machine-learning/concept-differential-privacy