Managed Services for Machine Learning

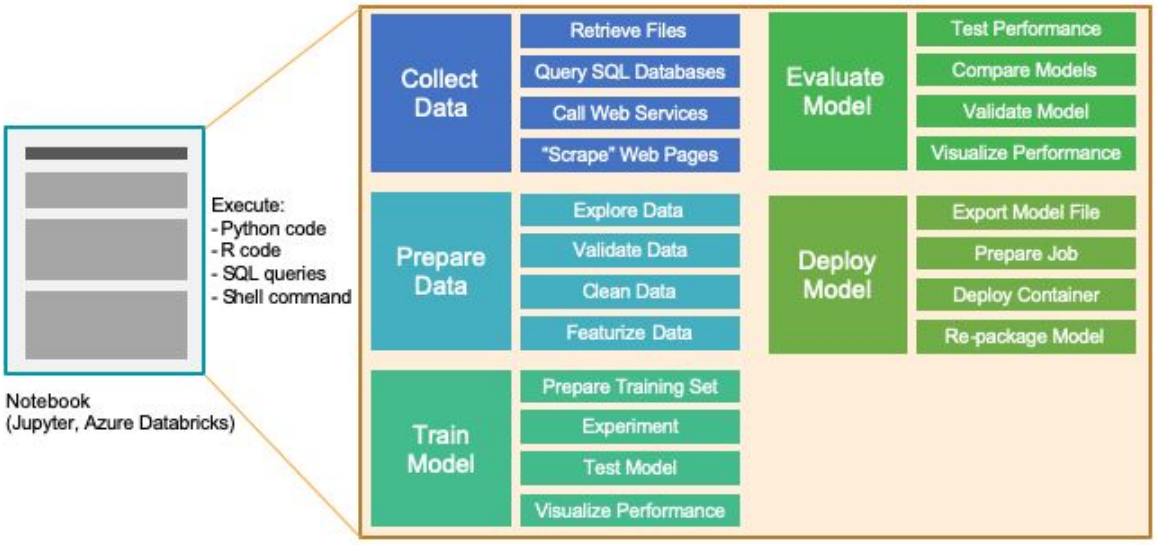
- The machine learning process can be labor intensive. Machine learning requires a number of tools to prepare the data, train the models, and deploy the models. Most of the work usually takes place within web-based, interactive notebooks, such as Jupyter notebooks. Although notebooks are lightweight and easily run in a web browser, you still need a server to host them. Typically, this involves installing several applications and libraries on a machine, configuring the environment settings, and then loading any additional resources required to begin working within notebooks or integrated development environments (IDEs).
- All this setup takes time, and there is sometimes a fair amount of troubleshooting involved to make sure you have the right combination of software versions that are compatible with one another. This is the advantage of **managed services for machine learning**, which provide a ready-made environment that is pre-optimized for your machine learning development.

Compute Resources

- Compute Resource is a service that provides virtual equipment (Compute Resources) by combining CPUs, Memory, and Disks to create Virtual Machines. Compute Resources are provided by virtualizing physical servers and storage devices shared by multiple users.
- There are two different variations on compute targets: **training compute targets** and **inferencing compute targets**.

Managed Notebook Environments

- Notebooks are made up of one or more cells that allow for the execution of the code snippets or commands within those cells.
- They store commands and the results of running those commands.
- In this diagram, you can see that we can use a notebook environment to perform the five primary stages of model development:



Basic Modeling

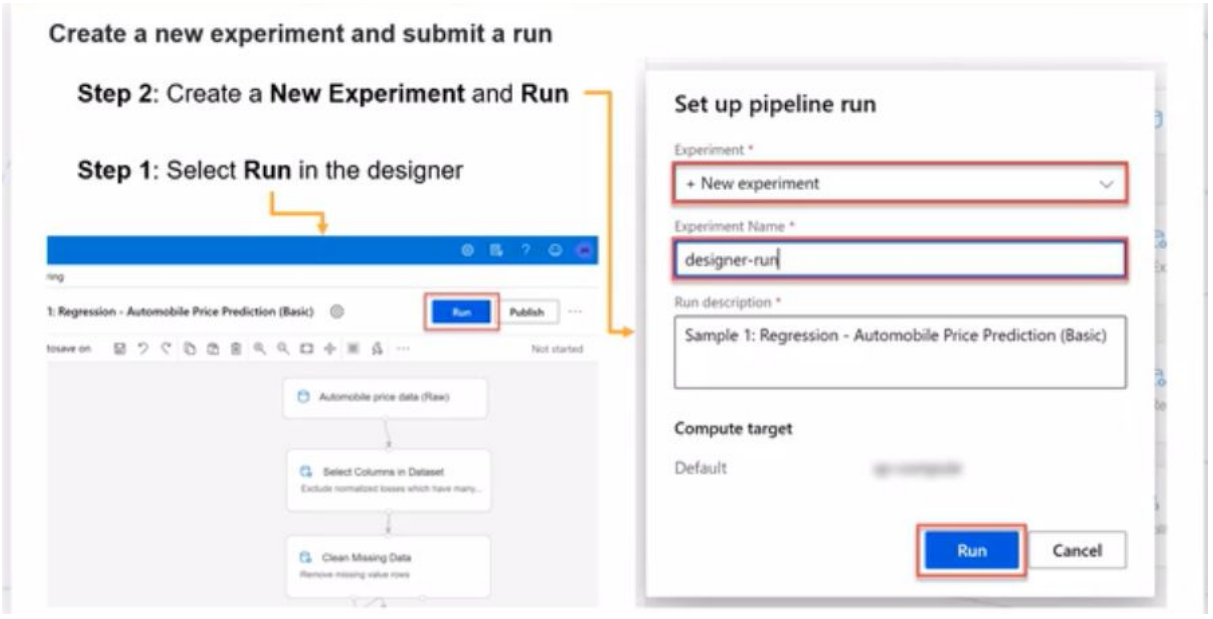
Training, evaluating, and selecting the right Machine Learning models is at the core of each modern data science process.
Basic Modelling consists of three steps:
1.Experiments
2.Runs
3.Model Registry

- ## Experiments

  An **experiment** is a generic context for handling runs. Think about it as a logical entity you can use to organize your model training processes.

❑ Runs

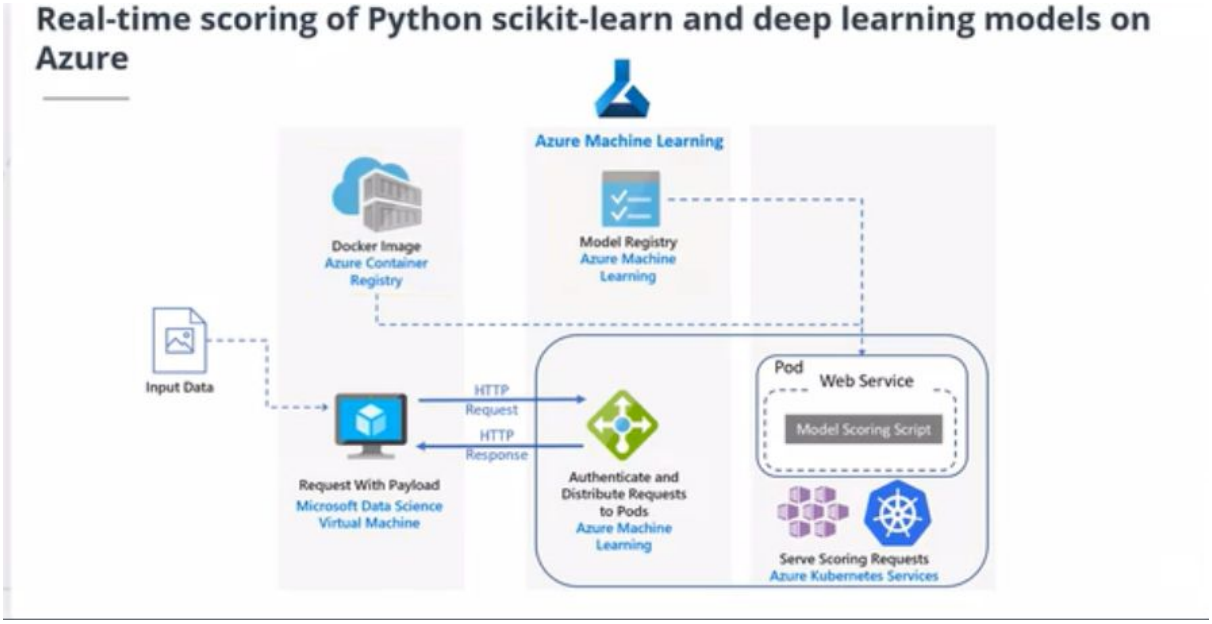
Once you have an experiment, you can create **runs** within that experiment. A run contains all artifacts associated with the training process, like output files, metrics, logs, and a snapshot of the directory that contains your scripts.



❑ Models

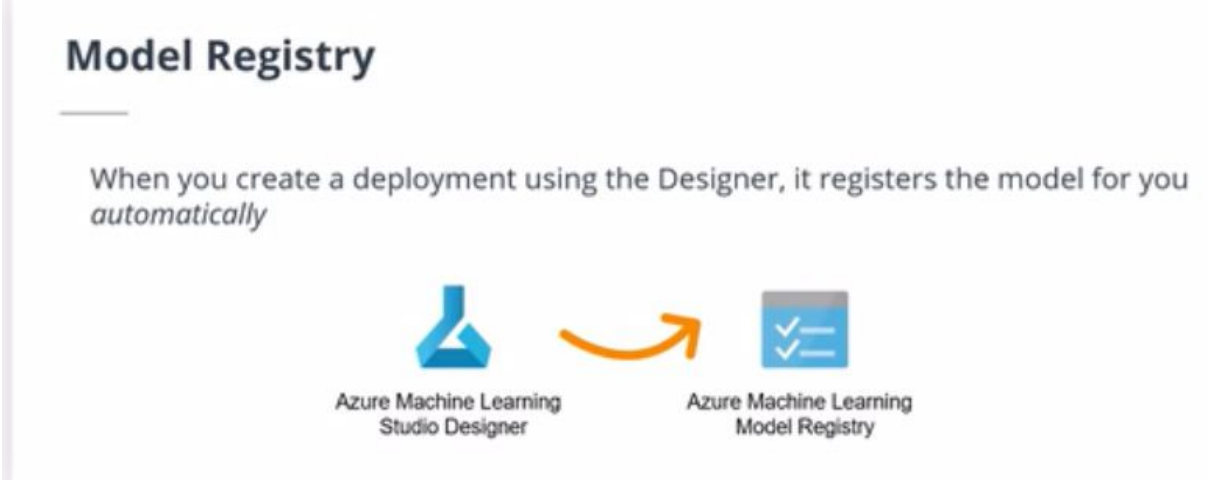
A run is used to produce a *model*. Essentially, a **model** is a piece of code that takes an input and produces output.

Model = algorithm + data + hyperparameters



❑ Model Registry

Once we have a trained model, we can turn to the **model registry**, which keeps track of all models in an Azure Machine Learning workspace.



Advanced Modeling

As the process of building your models becomes more complex, it becomes more important to get a handle on the steps to prepare your data and train your

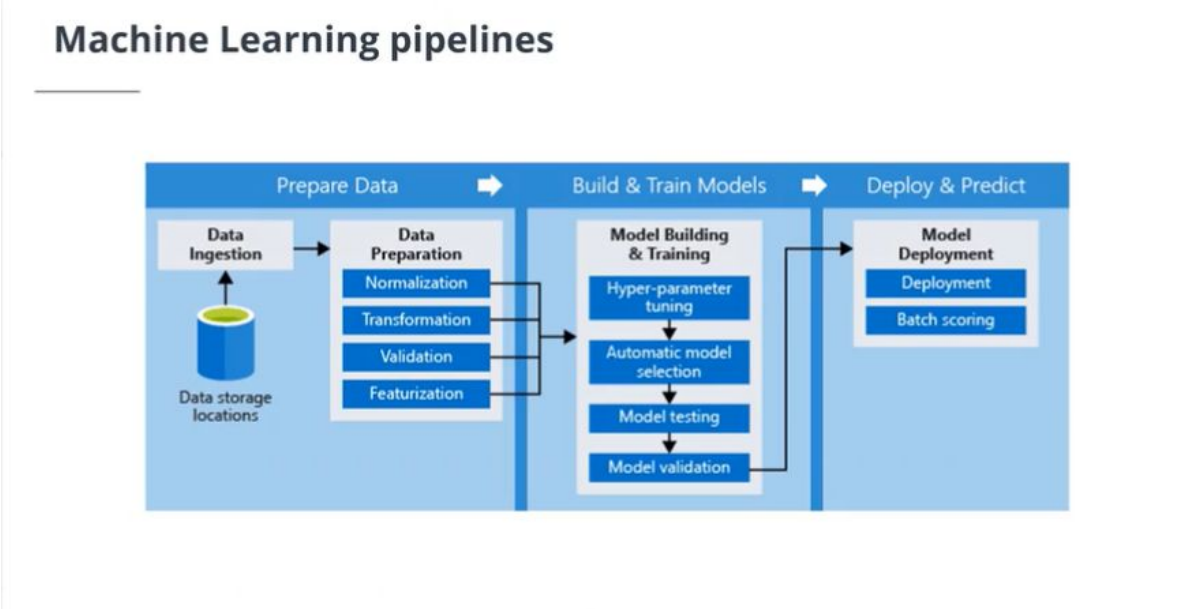
models in an organized way. In these scenarios, steps involved in the end-to-end process are:

- Data ingestion
- Data preparation
- Model building & training
- Model deployment.

Steps are organised into pipelines:

1.DevOps : process automation applied to classical software development

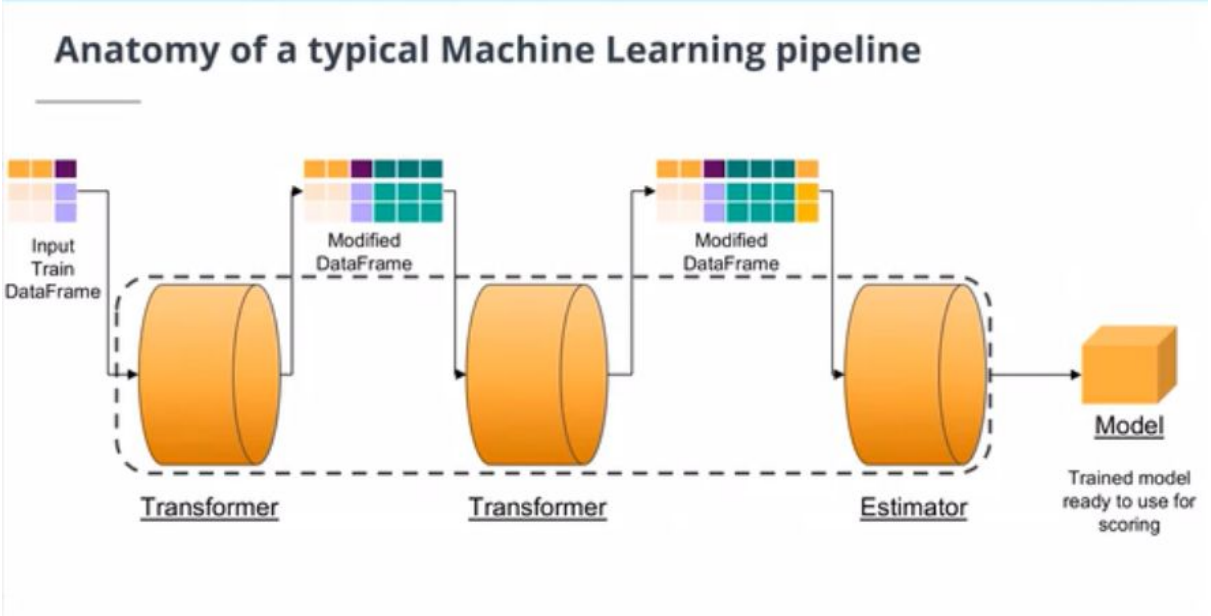
2.MLOps:applying DevOps principles to machine learning pipelines.



MLOps: Creating Automatic End-to-End Integrated Processes

- Model reproducibility
- Model Validation

- Model deployment
- Model retraining



Deploying a model-step by step

- Get a model file
- Create a scoring script(.py)
- Optionally create a schema file describing the web services input(.json)
- Create a real time web scoring service
- Call the web service from your applications
- Repeat the service each time you retrain the model

Real-time Inferencing

The model training process can be very compute-intensive, with training times that can potentially span across many hours, days, or even weeks. A trained model, on the other hand, is used to make decisions on new data quickly. In other words, it infers things about new data it is given based on its training. Making these decisions on new data on-demand is called **real-time inferencing**.

Deploying with Azure Machine Learning

- Trained model
- Use the trained model to make predictions

- Azure ML deploys the containers to Azure Kubernetes Service or Azure Container Instance.

Batch Inferencing

Unlike real-time inferencing, which makes predictions on data as it is received, **batch inferencing** is run on large quantities (batches) of existing data.
Typically, batch inferencing is run on a recurring schedule against data stored in a database or other data store.
Batch Interferencing is used when

- No need for real-time
- Interferencing results can be persisted
- Post-processing or analysis on the prediction is needed
- Interferencing is complex