Lesson 3 Model Training

Lesson Overview

Before training a model, we first need to handle **data preparation**, so we'll explore this topic first. More specifically, we will go over:

- Data importing and transformation
- The data management process, including:
    - The use of *datastores* and *datasets*
    - Versioning
    - Feature engineering
    - How to monitor for *data drift*

Next, we will introduce the basics of **model training**. We'll cover:

- The core model training process
- Two of the fundamental machine learning models: *Classifier* and *regressor*
- The model evaluation process and relevant metrics

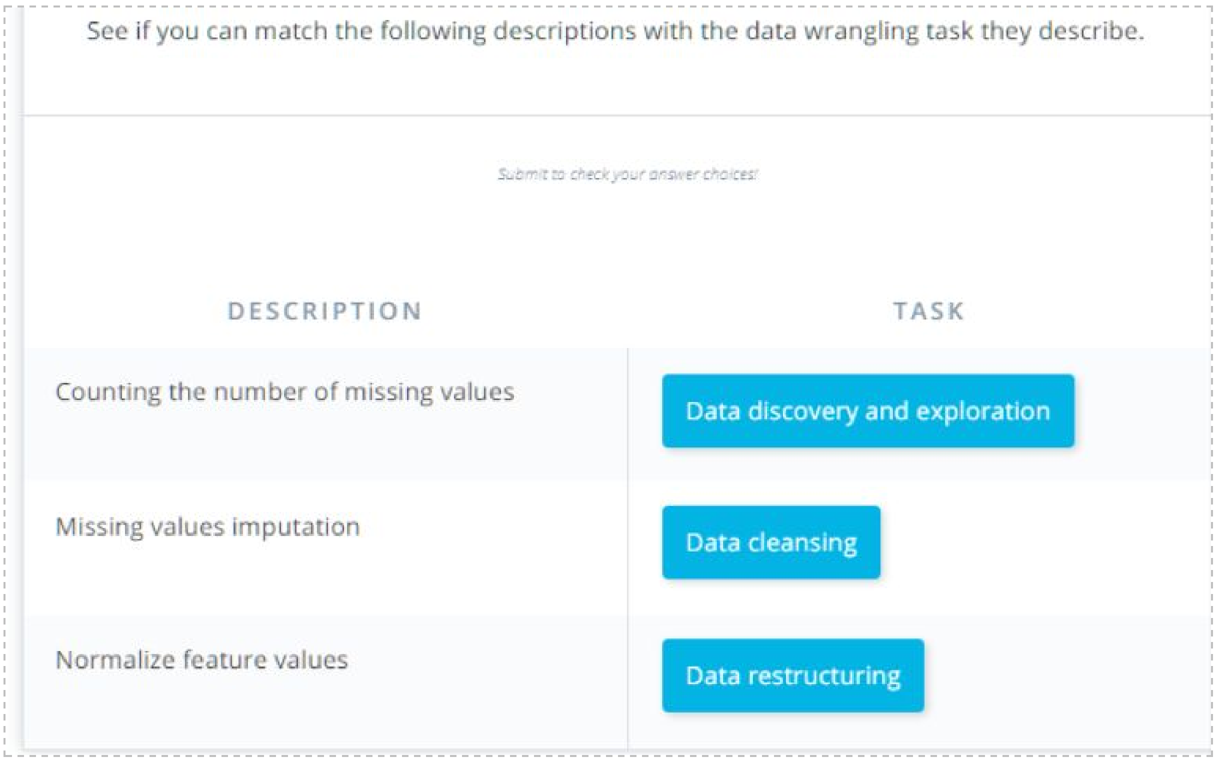
And finally, we'll conclude with an introduction to **ensemble learning** and **automated machine learning**, two core techniques used to make decisions based on multiple—rather than single—trained models.

Data Import and Transformation

Data wrangling is the process of cleaning and transforming data to make it more appropriate for data analysis. The process generally follows these main steps:

- Explore the raw data and check the general quality of the dataset.
- Transform the raw data, by restructuring, normalizing, and cleaning the data. For example, this could involve handling missing values and detecting errors.
- Validate and publish the data.

Data wrangling is an *iterative* process where you do some data transformation then check the results and come back to the process to make improvements.



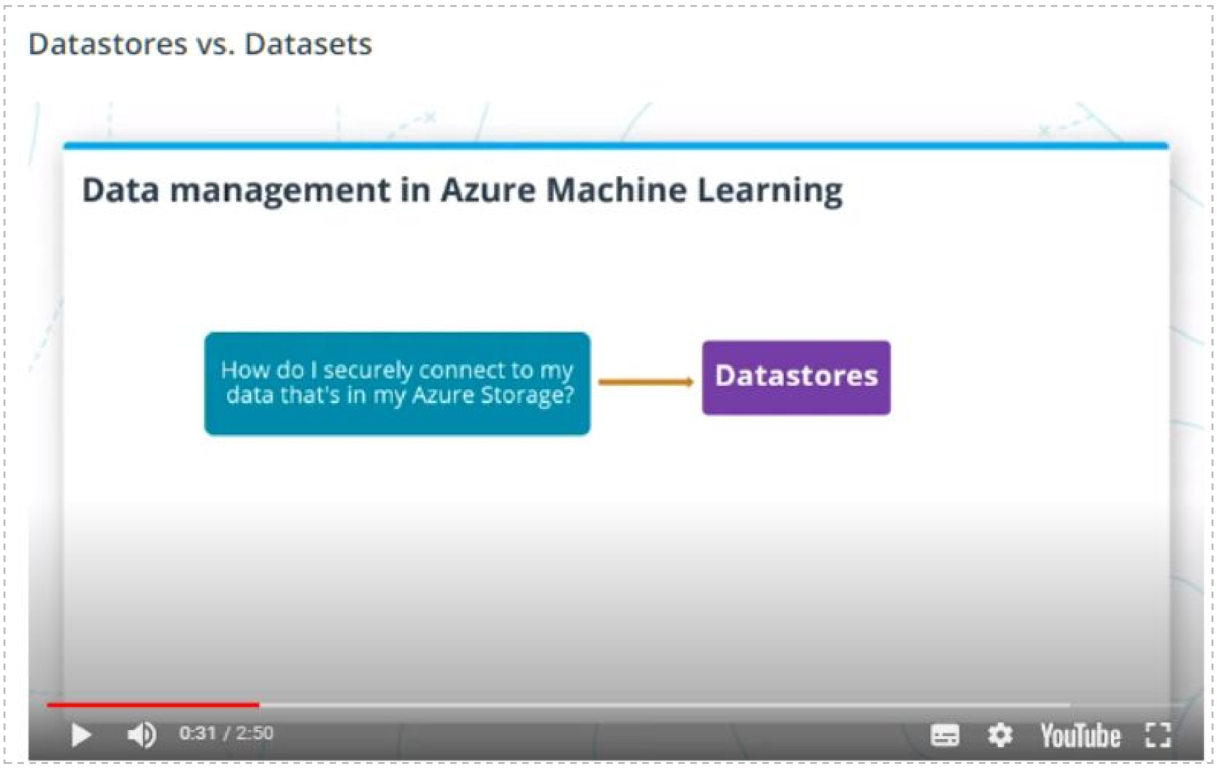
See if you can match the following descriptions with the data wrangling task they describe.

Submit to check your answer choices!

DESCRIPTION	TASK
Counting the number of missing values	Data discovery and exploration
Missing values imputation	Data cleansing
Normalize feature values	Data restructuring

Managing Data

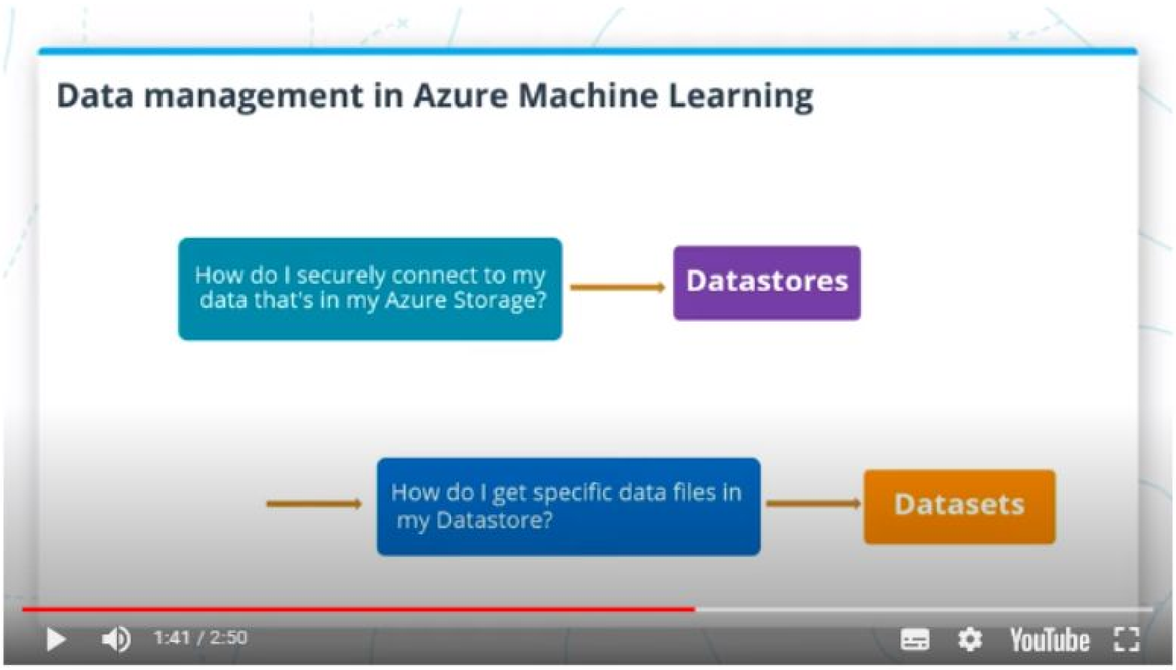
As we just discussed, Azure Machine Learning has two data management tools that we need to consider: **Datastores** and **datasets**. At first the distinction between the two may not be entirely clear, so let's have a closer look at what each one does and how they are related.


Datastores vs. Datasets

Data management in Azure Machine Learning

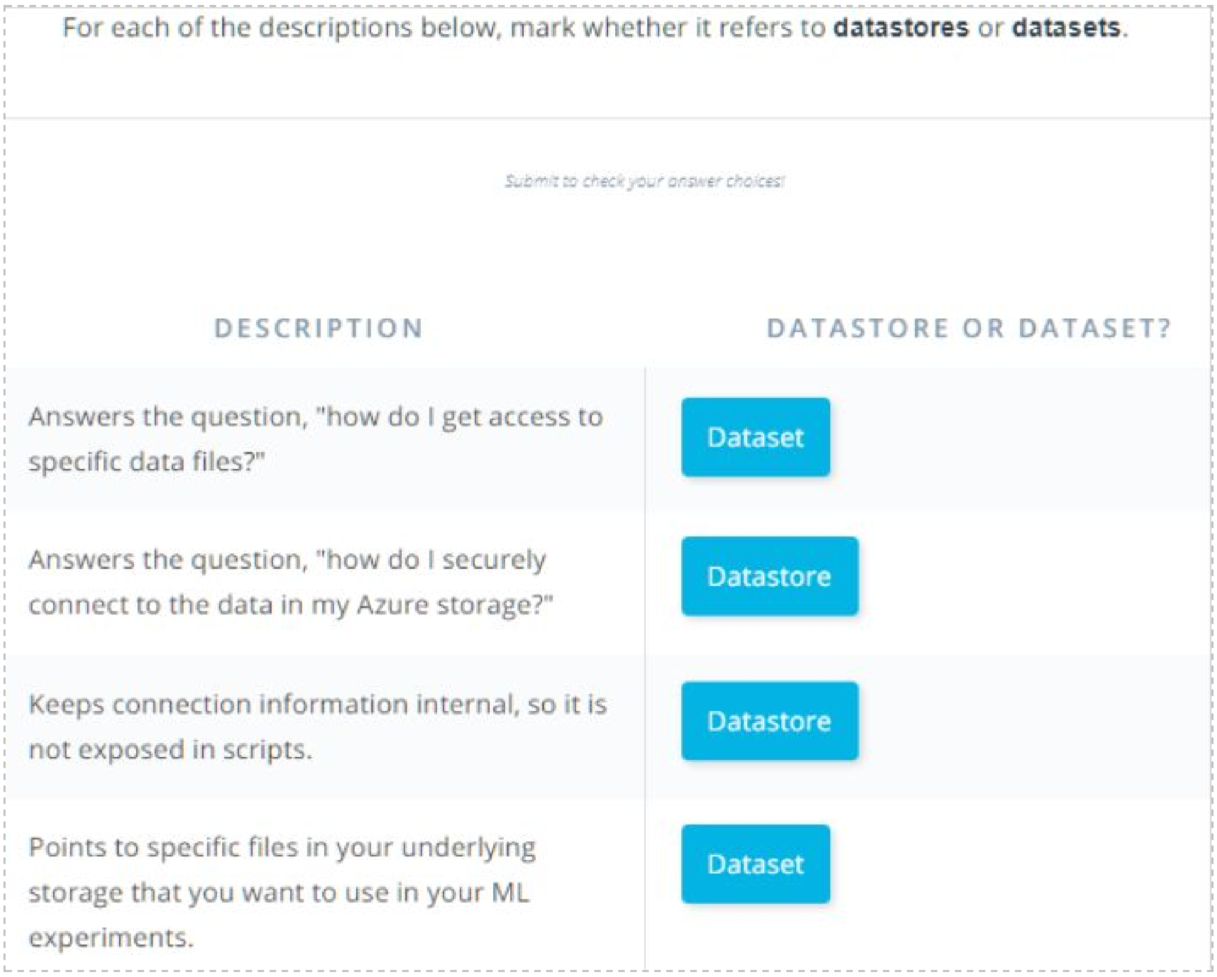
How do I securely connect to my data that's in my Azure Storage?
Datastores
0:31 / 2:50
YouTube

Datastores vs. Datasets


Data management in Azure Machine Learning
How do I securely connect to my data that's in my Azure Storage?
Datastores
How do I get specific data files in my Datastore?
Datasets
1:41 / 2:50
YouTube

Datastores offer a layer of abstraction over the supported Azure storage services. They store all the information needed to connect to a particular storage service. Datastores provide an access mechanism that is independent of the computer resource that is used to drive a machine learning process.

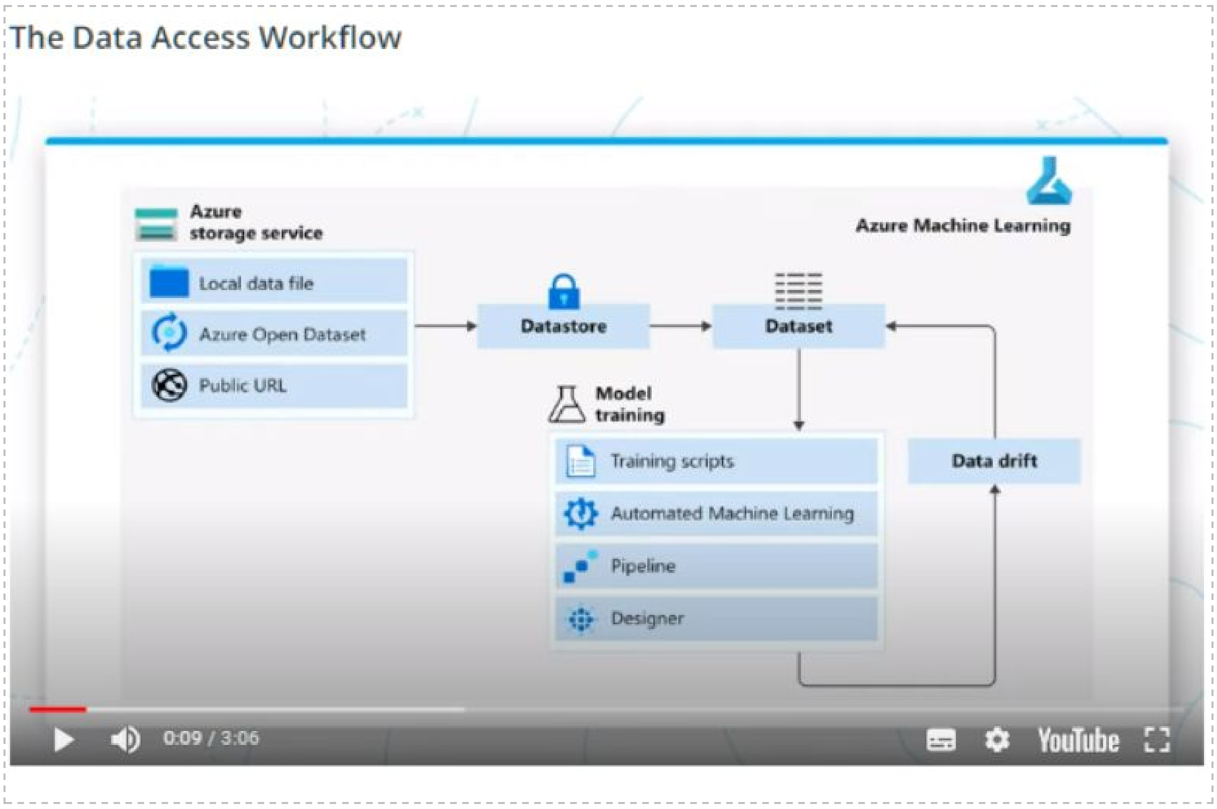
Datasets are resources for exploring, transforming, and managing data in Azure ML. A dataset is essentially a reference that points to the data in storage. It is used to get specific data files in the datastores.

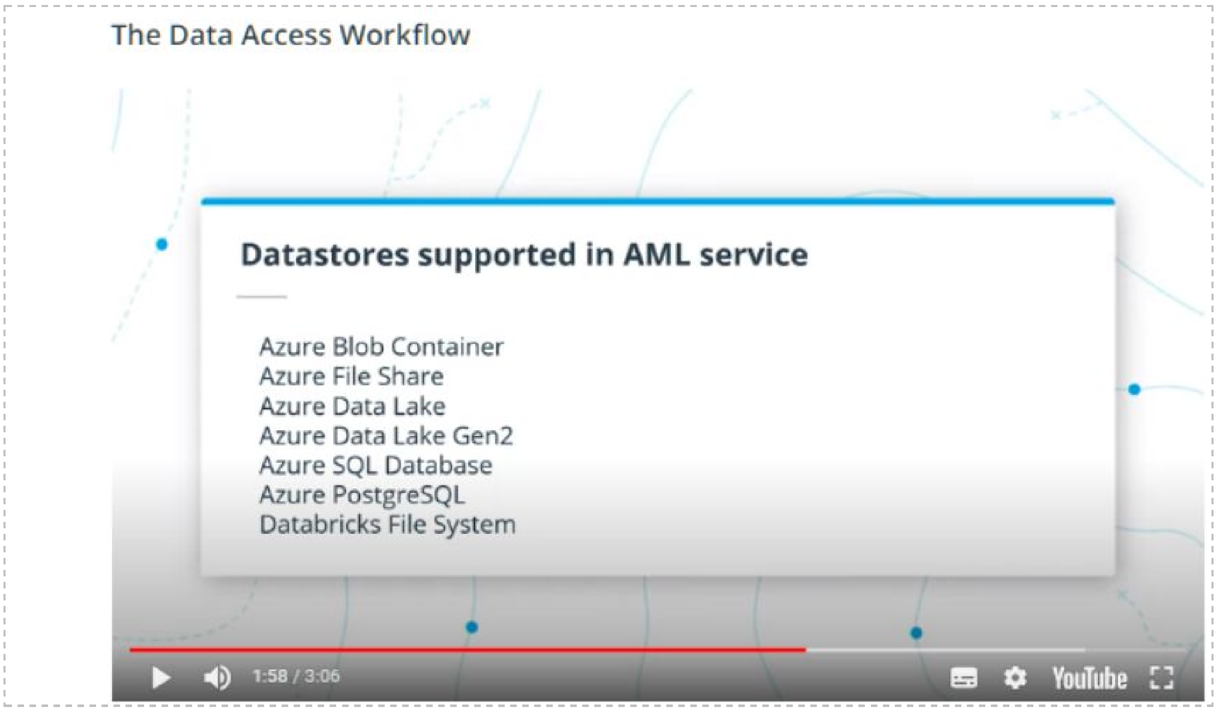

For each of the descriptions below, mark whether it refers to **datastores** or **datasets**.

Submit to check your answer choices!

DESCRIPTION	DATASTORE OR DATASET?
Answers the question, "how do I get access to specific data files?"	Dataset
Answers the question, "how do I securely connect to the data in my Azure storage?"	Datastore
Keeps connection information internal, so it is not exposed in scripts.	Datastore
Points to specific files in your underlying storage that you want to use in your ML experiments.	Dataset

The Data Access Workflow




The Data Access Workflow

Datastores supported in AML service

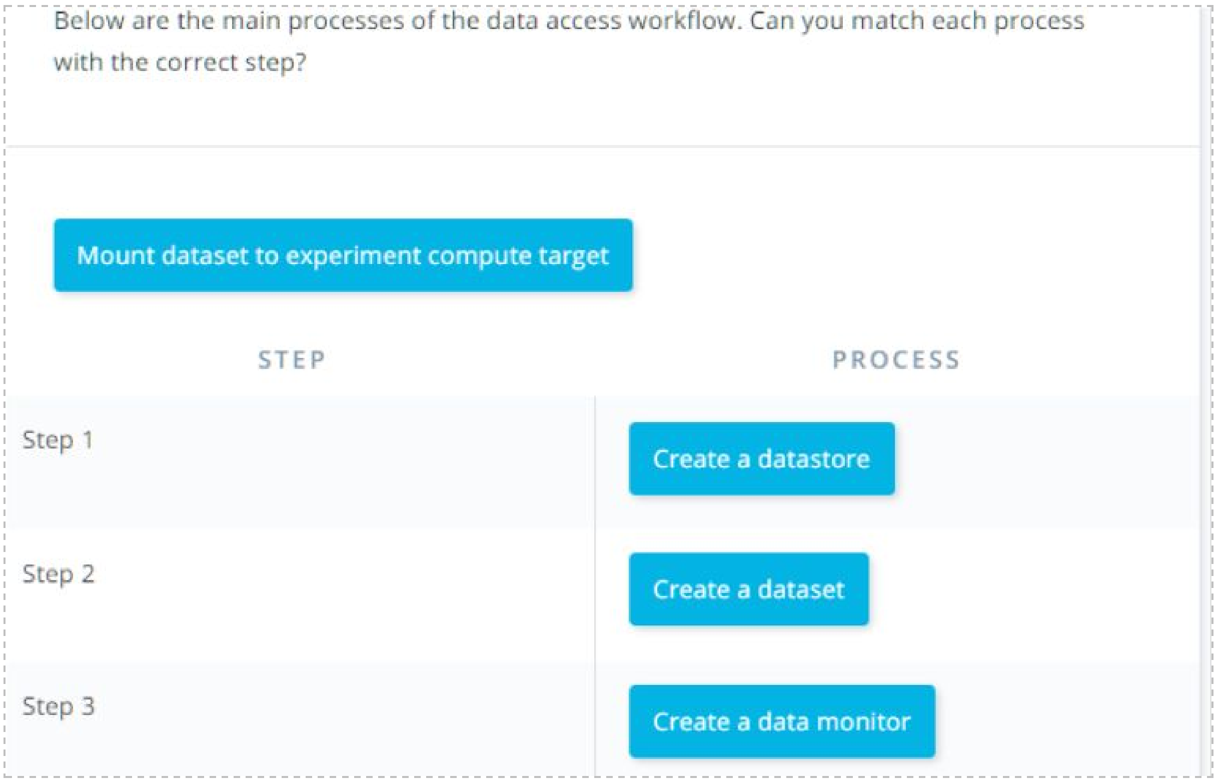
Azure Blob Container
Azure File Share
Azure Data Lake
Azure Data Lake Gen2
Azure SQL Database
Azure PostgreSQL
Databricks File System

The steps of the data access workflow are:

1. **Create a datastore** so that you can access storage services in Azure.
2. **Create a dataset**, which you will subsequently use for model training in your machine learning experiment.
3. **Create a dataset monitor** to detect issues in the data, such as data drift.

In the video, we mentioned the concept of *data drift*. Over time, the input data that you are feeding into your model is likely to change—and this is what we mean by **data drift**. Data drift can be problematic for model accuracy. Since you trained the model on a certain set of data, it can become increasingly inaccurate and the data changes more and more over time. For example, if you train a model to detect spam in email, it may become less accurate as new types of spam arise that are different from the spam on which the model was trained.

As we noted in the video, you can set up *dataset monitors* to detect data drift and other issues in your data. When data drift is detected, you can have the system automatically update

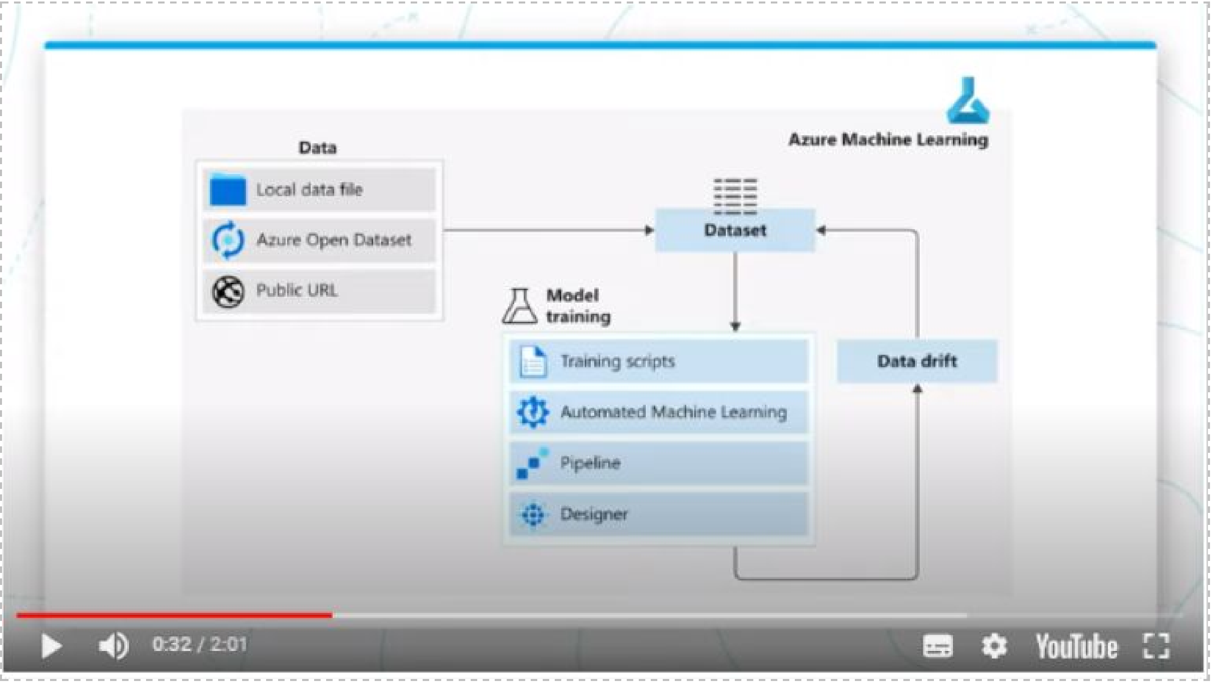

Below are the main processes of the data access workflow. Can you match each process with the correct step?

Mount dataset to experiment compute target

STEP	PROCESS
Step 1	Create a datastore
Step 2	Create a dataset
Step 3	Create a data monitor

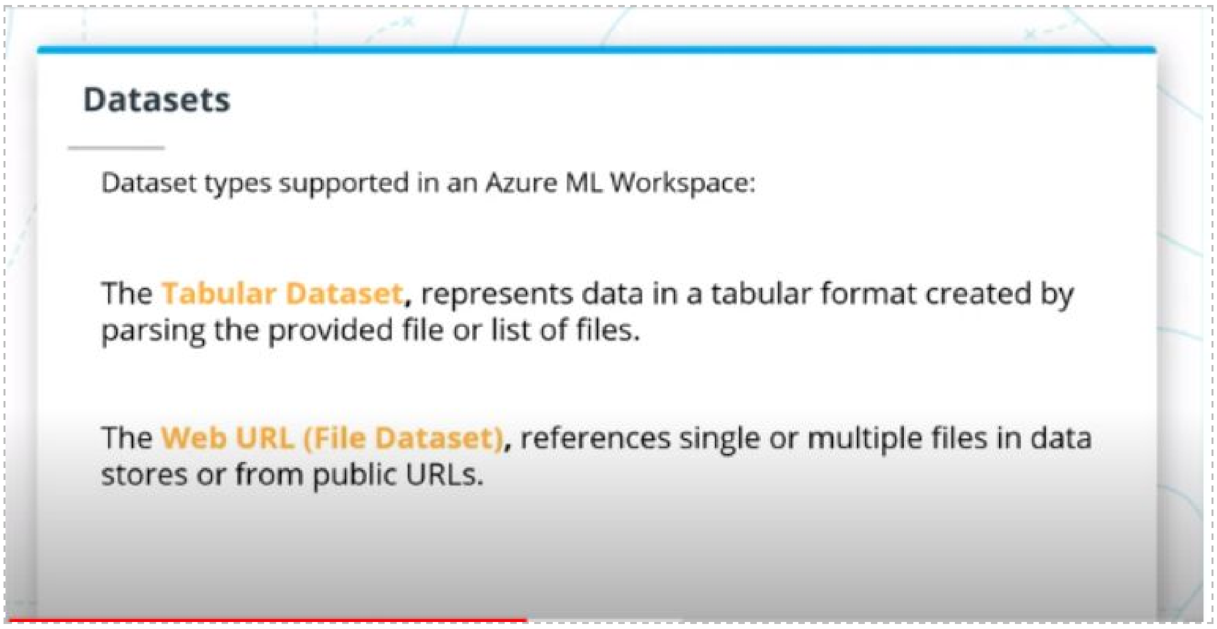
More About Datasets

On the last page, we discussed the main features of *datastores* and *datasets*. Now, let's look a little more closely at how datasets work in Azure Machine Learning.



Key points to remember about datasets:

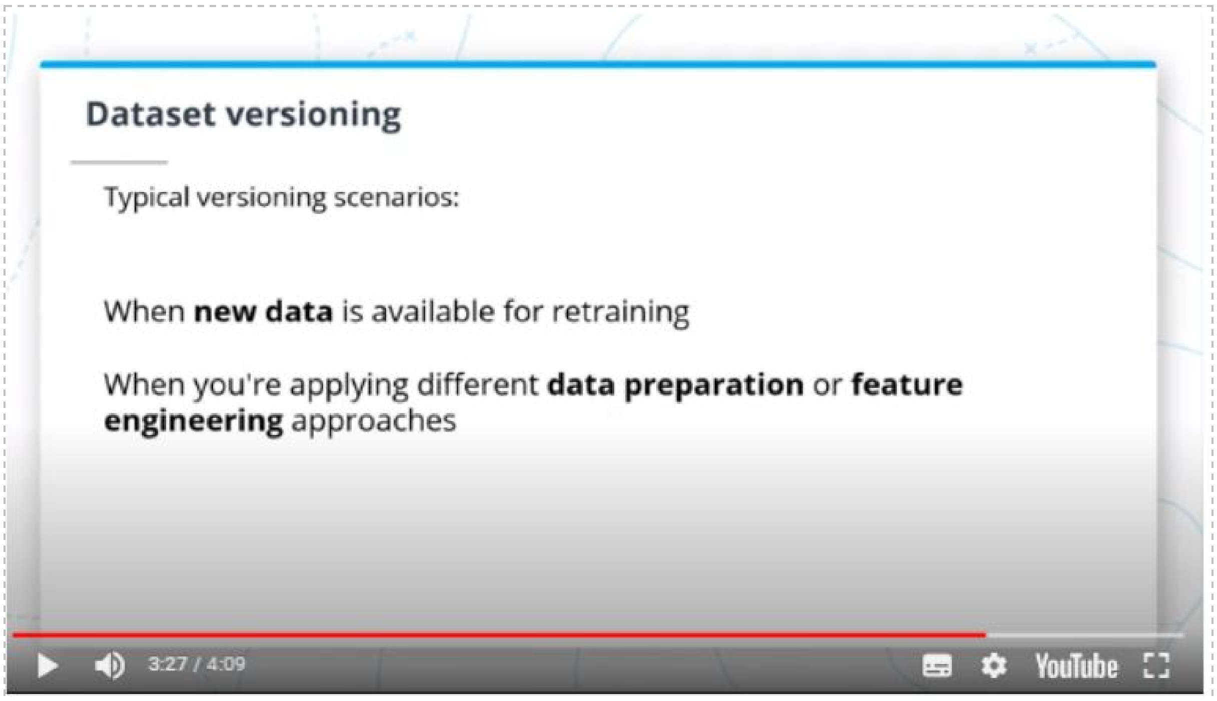
- They are used to interact with your data in the datastore and to package data into consumable objects.
- They can be created from local files, public URLs, Azure Open Datasets, and files uploaded to the datastores.
- They are not copies of the data but *references* that point to the original data. This means that no extra storage cost is incurred when you create a new dataset.
- Once a dataset is registered in Azure ML workspace, you can share it and reuse it across various other experiments without data ingestion complexities.


Datasets

Dataset types supported in an Azure ML Workspace:

The **Tabular Dataset**, represents data in a tabular format created by parsing the provided file or list of files.

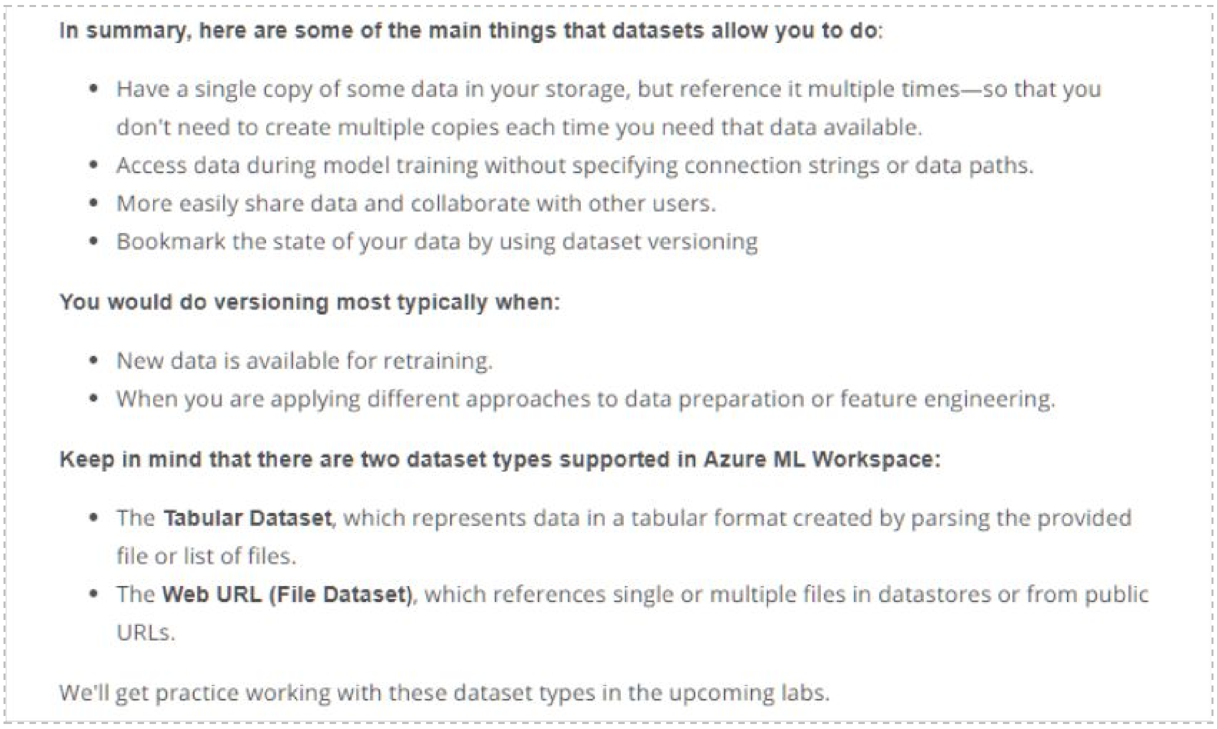
The **Web URL (File Dataset)**, references single or multiple files in data stores or from public URLs.


Dataset versioning

Typical versioning scenarios:

When **new data** is available for retraining

When you're applying different **data preparation** or **feature engineering** approaches


In summary, here are some of the main things that datasets allow you to do:

- Have a single copy of some data in your storage, but reference it multiple times—so that you don't need to create multiple copies each time you need that data available.
- Access data during model training without specifying connection strings or data paths.
- More easily share data and collaborate with other users.
- Bookmark the state of your data by using dataset versioning

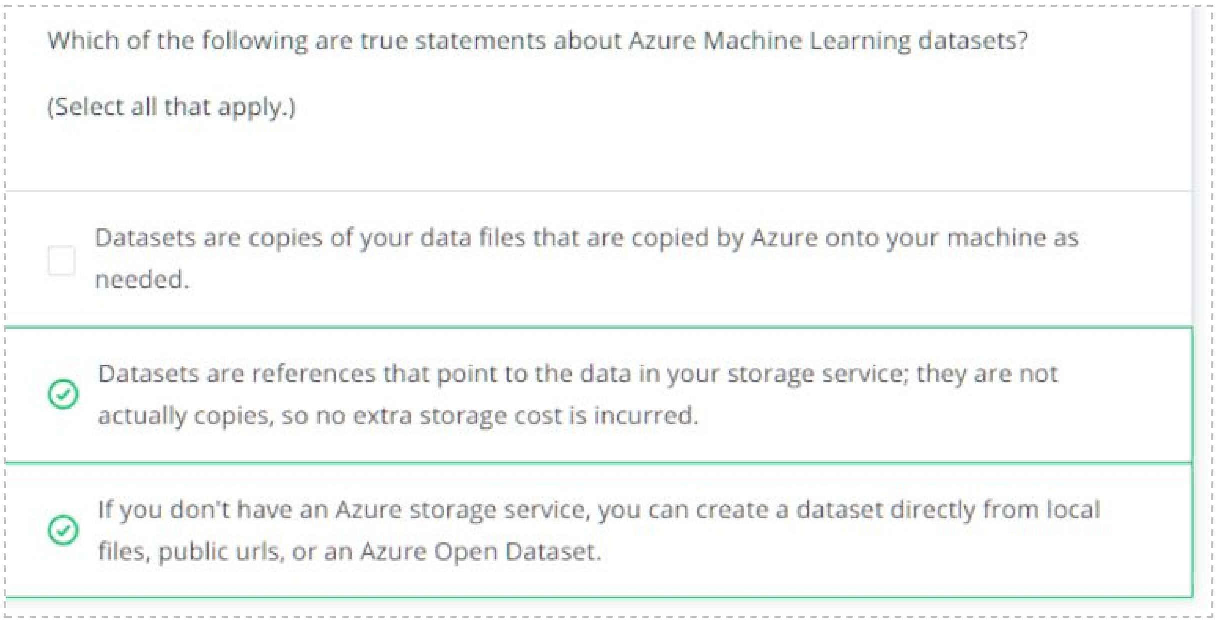
You would do versioning most typically when:

- New data is available for retraining.
- When you are applying different approaches to data preparation or feature engineering.

Keep in mind that there are two dataset types supported in Azure ML Workspace:

- The **Tabular Dataset**, which represents data in a tabular format created by parsing the provided file or list of files.
- The **Web URL (File Dataset)**, which references single or multiple files in datastores or from public URLs.

We'll get practice working with these dataset types in the upcoming labs.


Which of the following are true statements about Azure Machine Learning datasets?

(Select all that apply.)

- [ ] Datasets are copies of your data files that are copied by Azure onto your machine as needed.
- [✓] Datasets are references that point to the data in your storage service; they are not actually copies, so no extra storage cost is incurred.
- [✓] If you don't have an Azure storage service, you can create a dataset directly from local files, public urls, or an Azure Open Dataset.

Introducing Features

In the previous lesson, we took a look at some examples of *tabular data*:

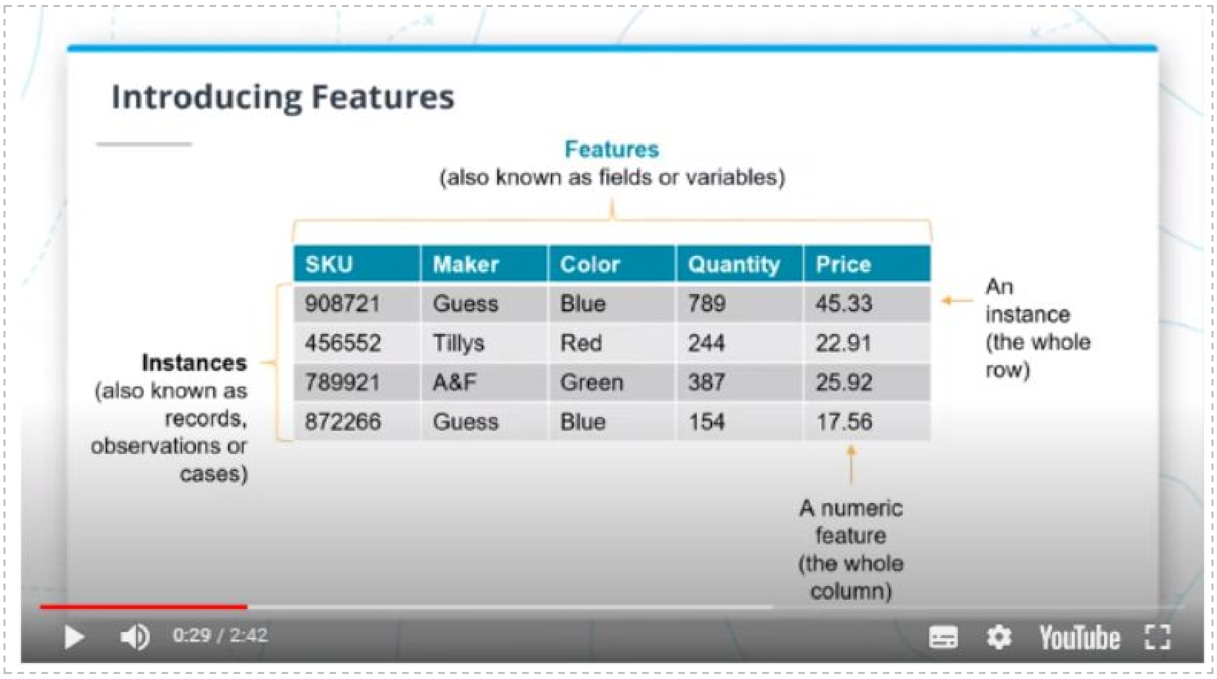
SKU	Make	Color	Quantity	Price
908721	Guess	Blue	789	45.33
456552	Tillys	Red	244	22.91
789921	A&F	Green	387	25.92
872266	Guess	Blue	154	17.56

> ***Note:*** *We have been referring to this as a data table, but you will also see data in this format called a* ***matrix****. The term matrix is commonly used in mathematics, and it refers to a rectangular array—which, just like a table, contains data arranged in rows and columns.*

Recall that the columns in a table can be referred to as **features**. In the above example, `color` and `quantity` are *features* of the products. In the last lesson, we mentioned briefly that **feature engineering** is an important part of data preparation. In this lesson, we'll look at this topic in more detail.

Introducing Features

SKU	Maker	Color	Quantity	Price
908721	Guess	Blue	789	45.33
456552	Tillys	Red	244	22.91
789921	A&F	Green	387	25.92
872266	Guess	Blue	154	17.56



Introducing Features

Features
(also known as fields or variables)

SKU	Maker	Color	Quantity	Price
908721	Guess	Blue	789	45.33
456552	Tillys	Red	244	22.91
789921	A&F	Green	387	25.92
872266	Guess	Blue	154	17.56

Instances (also known as records, observations or cases)

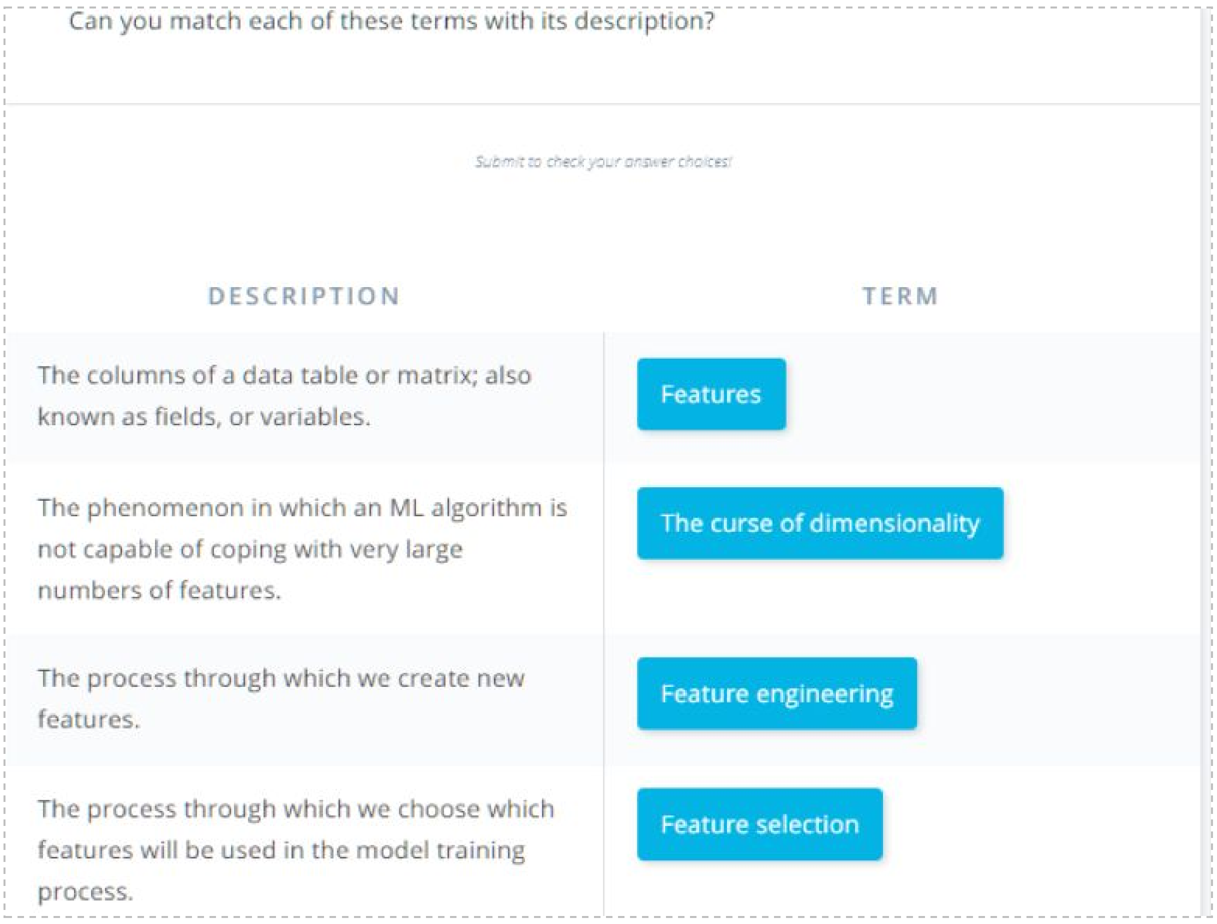
An instance (the whole row)

A numeric feature (the whole column)

In many cases, the set of initial features in the data is not enough to produce high quality trained machine learning models. You can use **feature engineering** to derive new features based on the values of existing features. This process can be as simple as applying a mathematical function to a feature (such as adding 1 to all values in an existing feature) or it can be as complex as training a separate machine learning model to create values for new features.

Once you have the features, another important task is selecting the features that are most important or most relevant. This process is called **feature selection**.

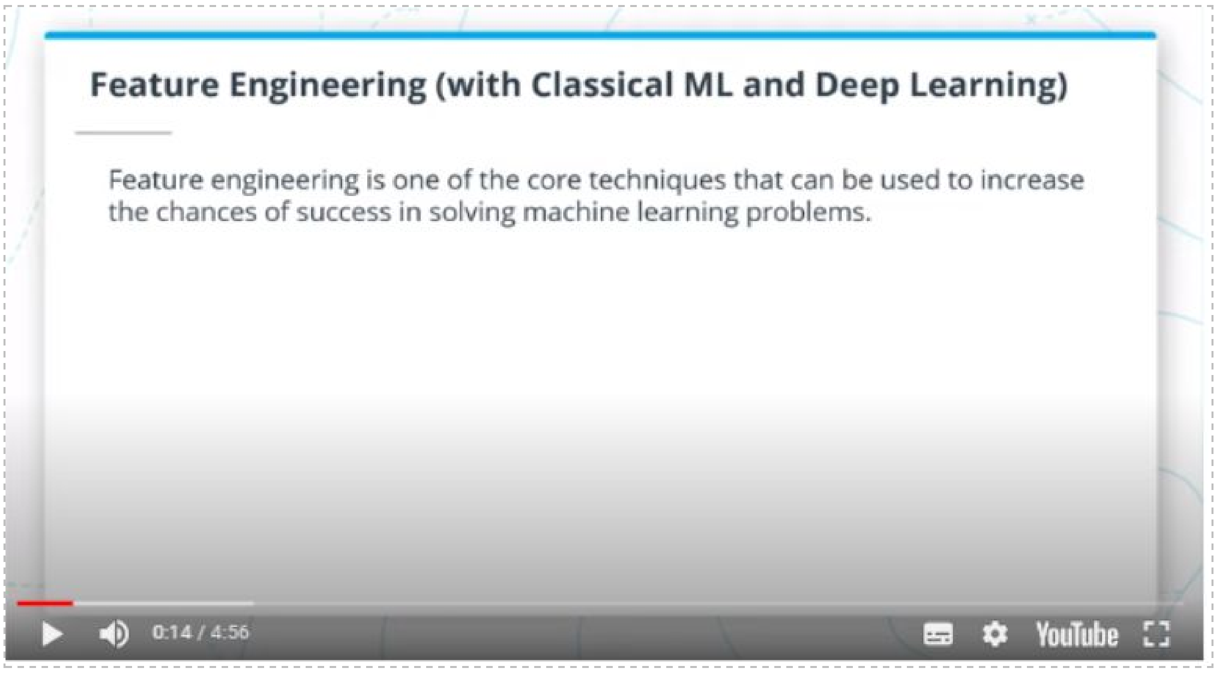
Many machine learning algorithms cannot accommodate a large number of features, so it is often necessary to do **dimensionality reduction** to decrease the number of features.


Can you match each of these terms with its description?

Submit to check your answer choices!

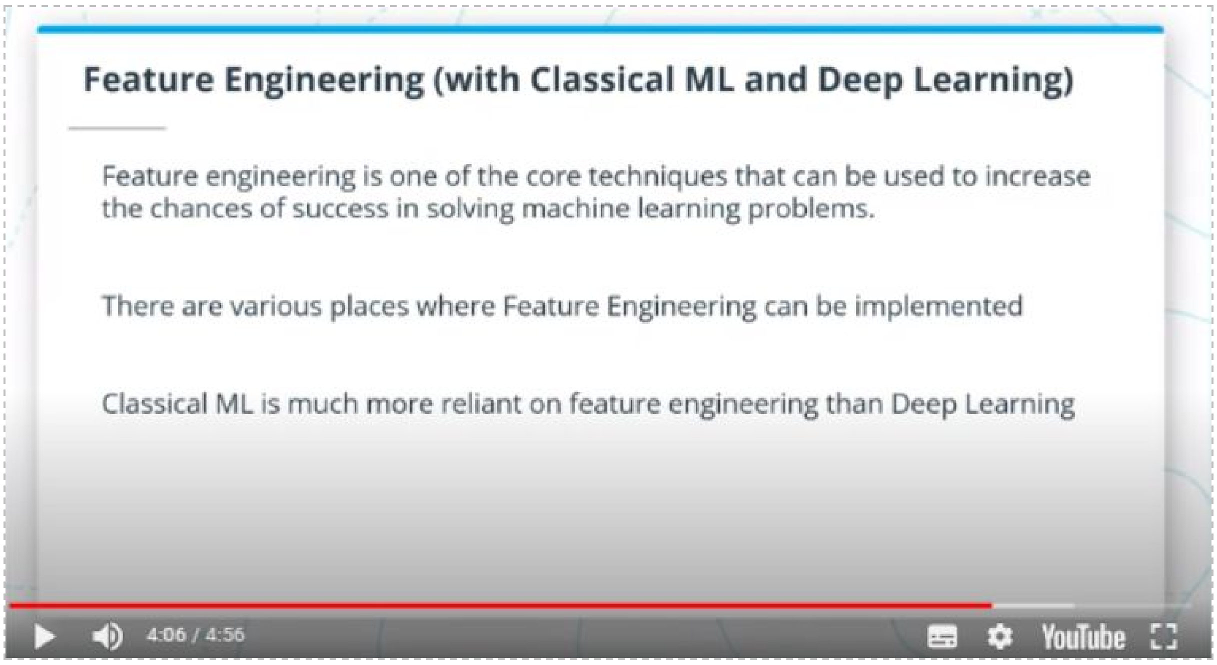
DESCRIPTION	TERM
The columns of a data table or matrix; also known as fields, or variables.	Features
The phenomenon in which an ML algorithm is not capable of coping with very large numbers of features.	The curse of dimensionality
The process through which we create new features.	Feature engineering
The process through which we choose which features will be used in the model training process.	Feature selection

Feature Engineering



Feature Engineering (with Classical ML and Deep Learning)

Feature engineering is one of the core techniques that can be used to increase the chances of success in solving machine learning problems.

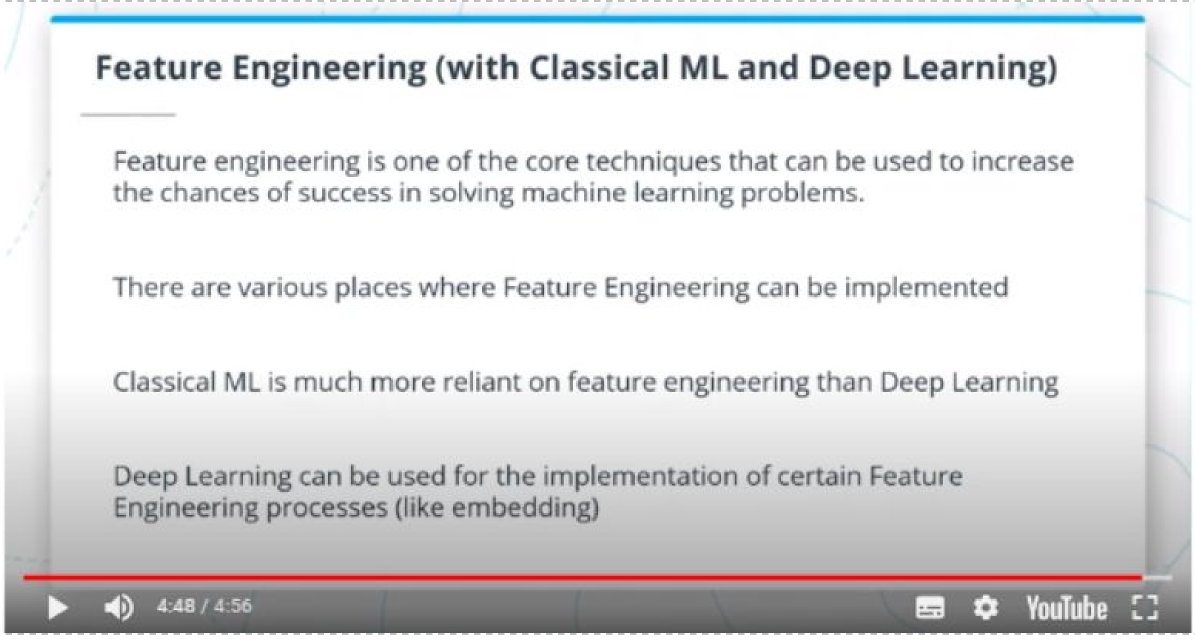


Feature Engineering (with Classical ML and Deep Learning)

Feature engineering is one of the core techniques that can be used to increase the chances of success in solving machine learning problems.

There are various places where Feature Engineering can be implemented

Classical ML is much more reliant on feature engineering than Deep Learning



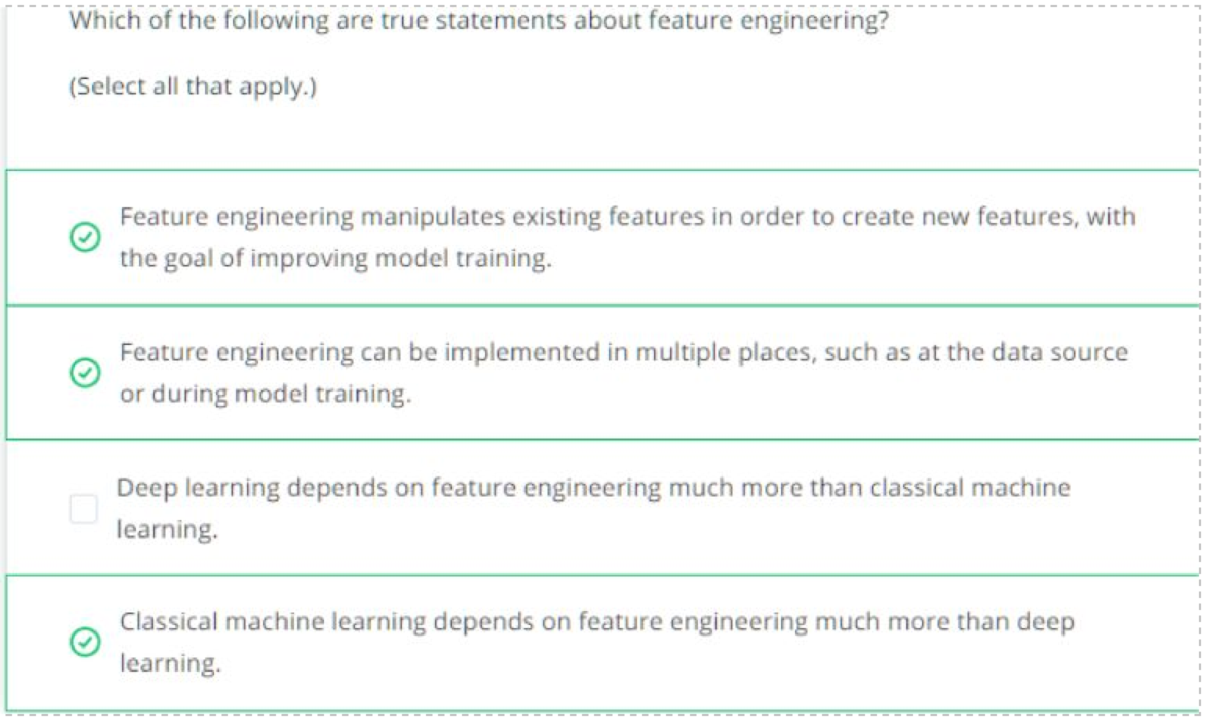
Feature Engineering (with Classical ML and Deep Learning)

Feature engineering is one of the core techniques that can be used to increase the chances of success in solving machine learning problems.

There are various places where Feature Engineering can be implemented

Classical ML is much more reliant on feature engineering than Deep Learning

Deep Learning can be used for the implementation of certain Feature Engineering processes (like embedding)

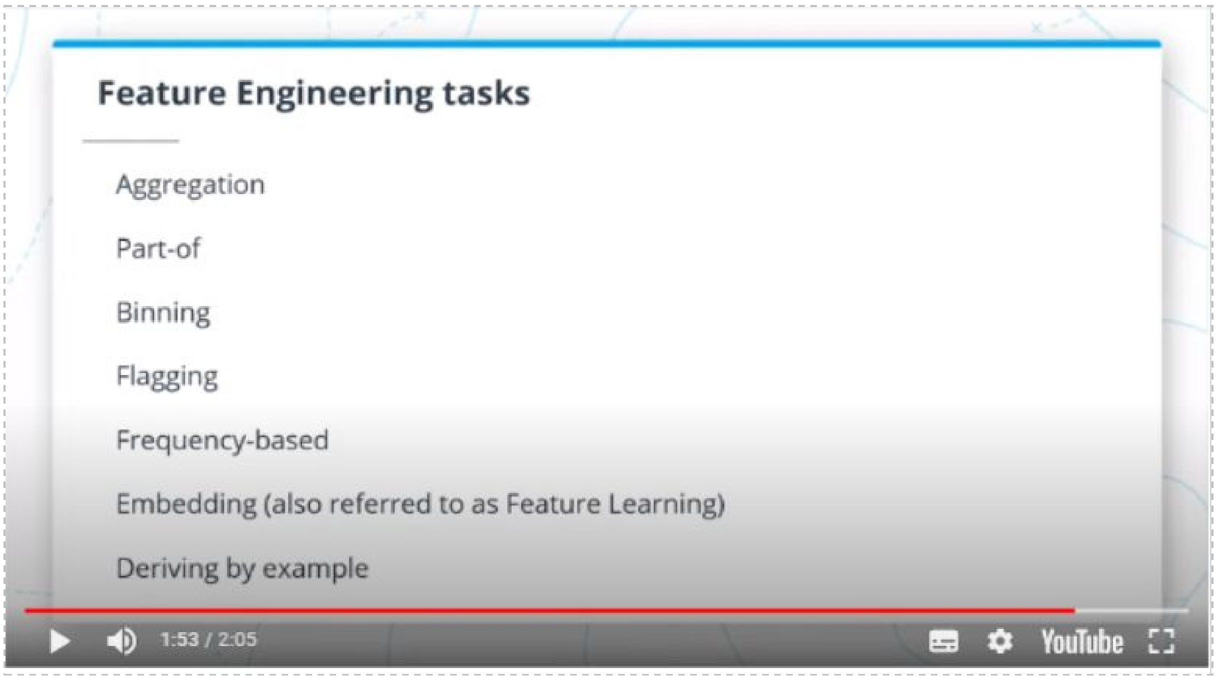


Which of the following are true statements about feature engineering?

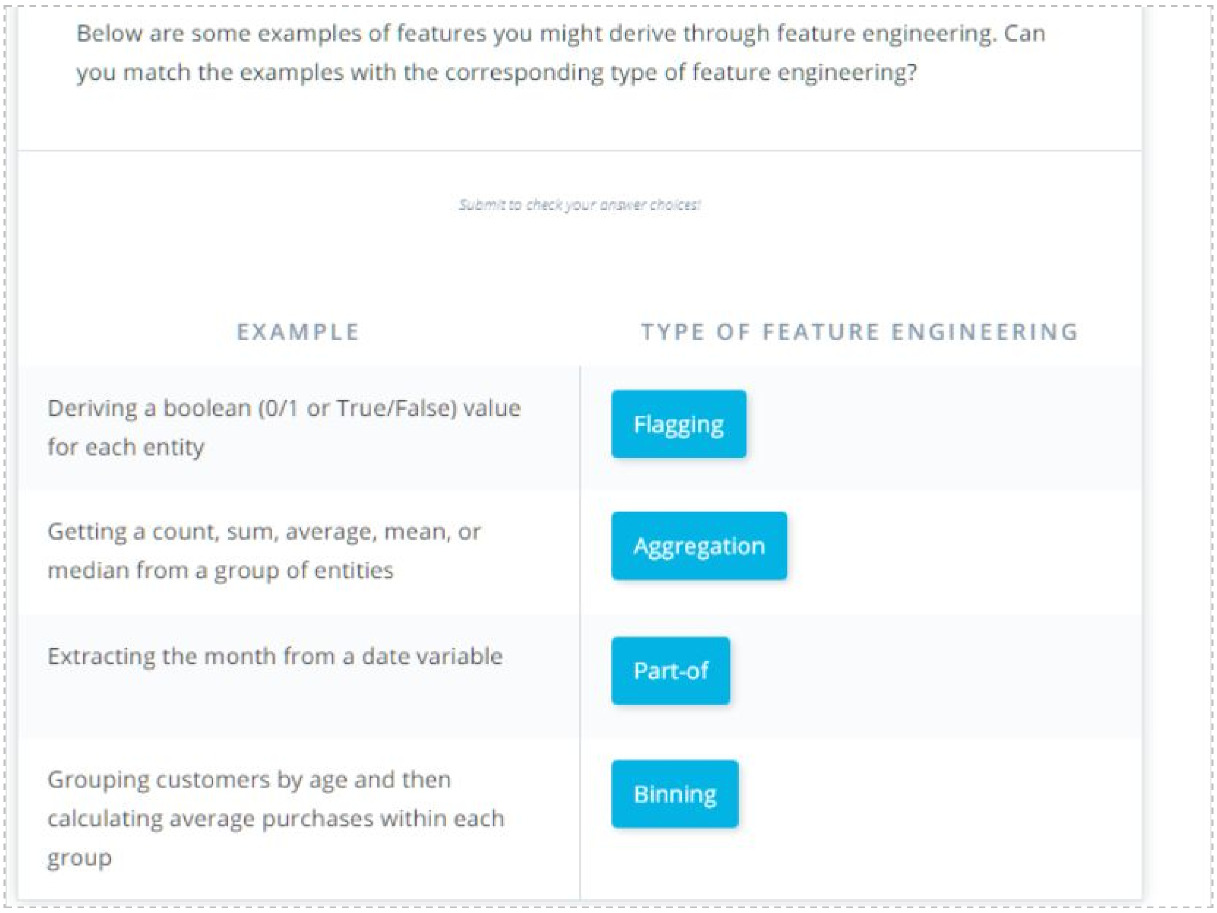
(Select all that apply.)

- ✓ Feature engineering manipulates existing features in order to create new features, with the goal of improving model training.
- ✓ Feature engineering can be implemented in multiple places, such as at the data source or during model training.
- ☐ Deep learning depends on feature engineering much more than classical machine learning.
- ✓ Classical machine learning depends on feature engineering much more than deep learning.

Examples of Feature Engineering Tasks


Feature Engineering tasks

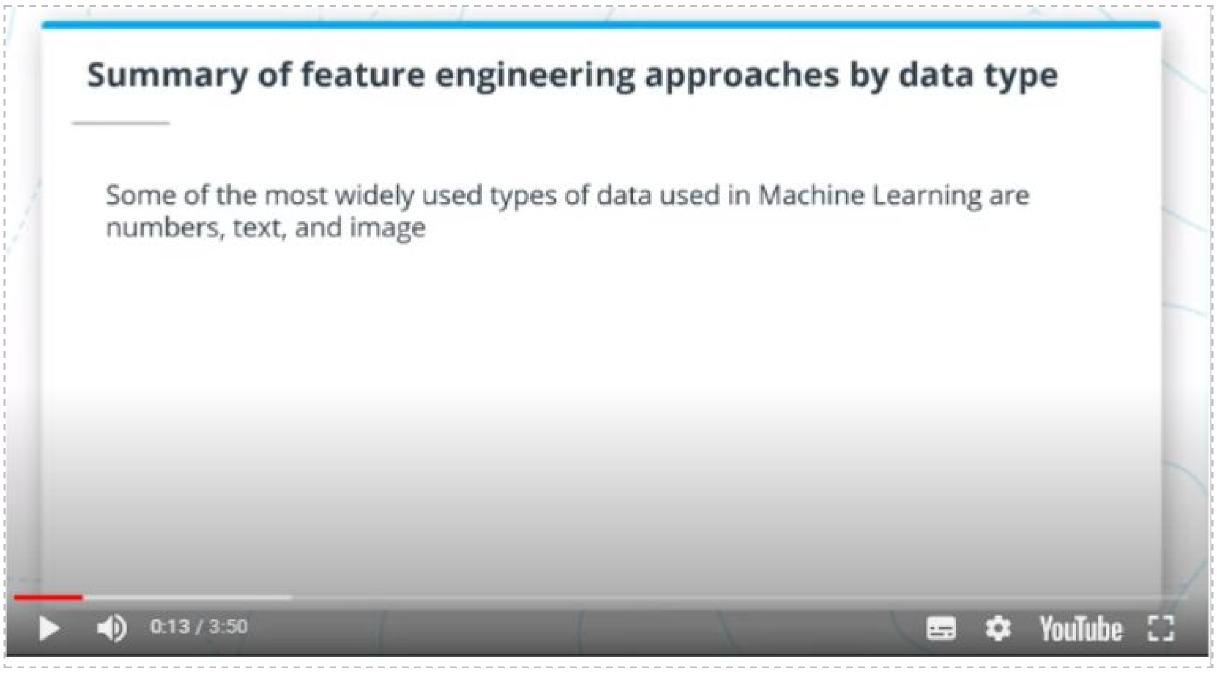
- Aggregation
- Part-of
- Binning
- Flagging
- Frequency-based
- Embedding (also referred to as Feature Learning)
- Deriving by example


Below are some examples of features you might derive through feature engineering. Can you match the examples with the corresponding type of feature engineering?

Submit to check your answer choices!

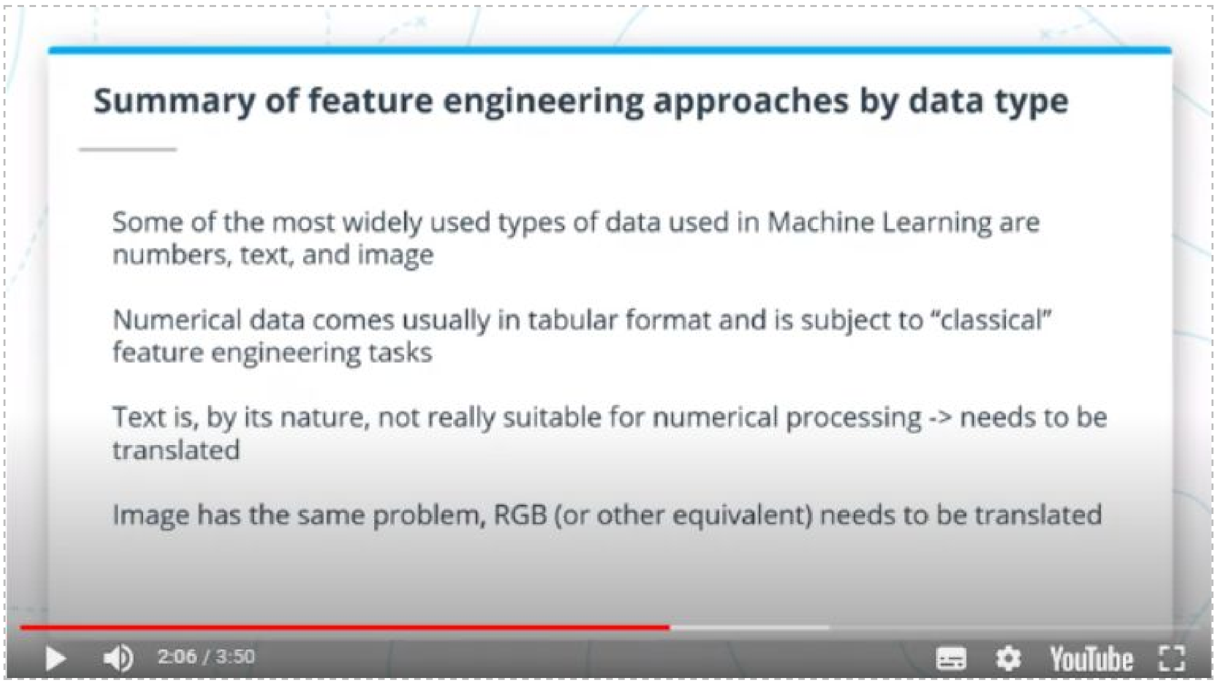
EXAMPLE	TYPE OF FEATURE ENGINEERING
Deriving a boolean (0/1 or True/False) value for each entity	Flagging
Getting a count, sum, average, mean, or median from a group of entities	Aggregation
Extracting the month from a date variable	Part-of
Grouping customers by age and then calculating average purchases within each group	Binning

Summary of Feature Engineering Approaches by Data Type



Summary of feature engineering approaches by data type

Some of the most widely used types of data used in Machine Learning are numbers, text, and image



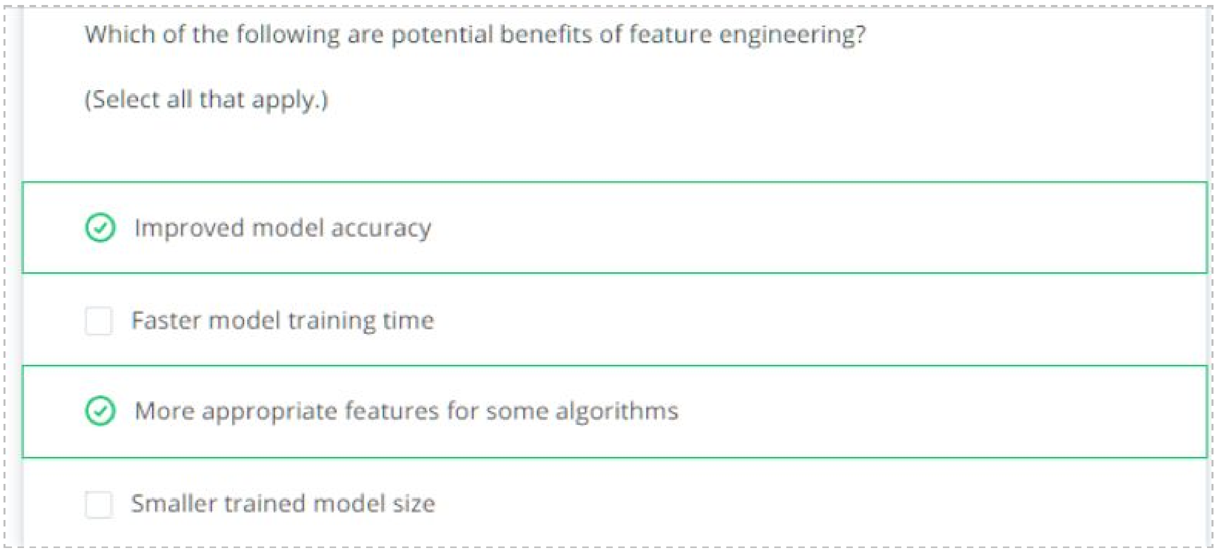
Summary of feature engineering approaches by data type

Some of the most widely used types of data used in Machine Learning are numbers, text, and image

Numerical data comes usually in tabular format and is subject to “classical” feature engineering tasks

Text is, by its nature, not really suitable for numerical processing -> needs to be translated

Image has the same problem, RGB (or other equivalent) needs to be translated

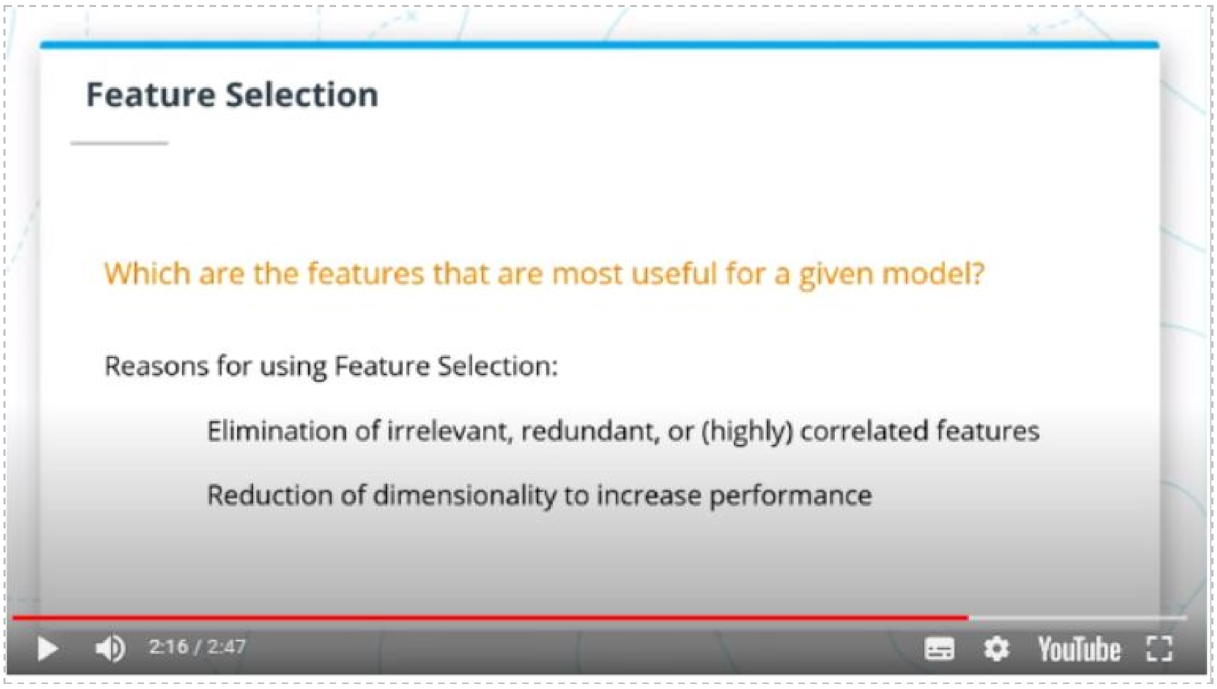

Which of the following are potential benefits of feature engineering?

(Select all that apply.)

- [✓ Improved model accuracy]
- Faster model training time
- [✓ More appropriate features for some algorithms]
- Smaller trained model size

Feature Selection

There are mainly two reasons for feature selection. Some features might be highly irrelevant or redundant. So it's better to remove these features to simplify the situation and improve performance. Additionally, it may seem like engineering more features is always a good thing, but as we mentioned earlier, many machine learning algorithms suffer from the *curse of dimensionality*—that is, they do not perform well when given a large number of variables or features.

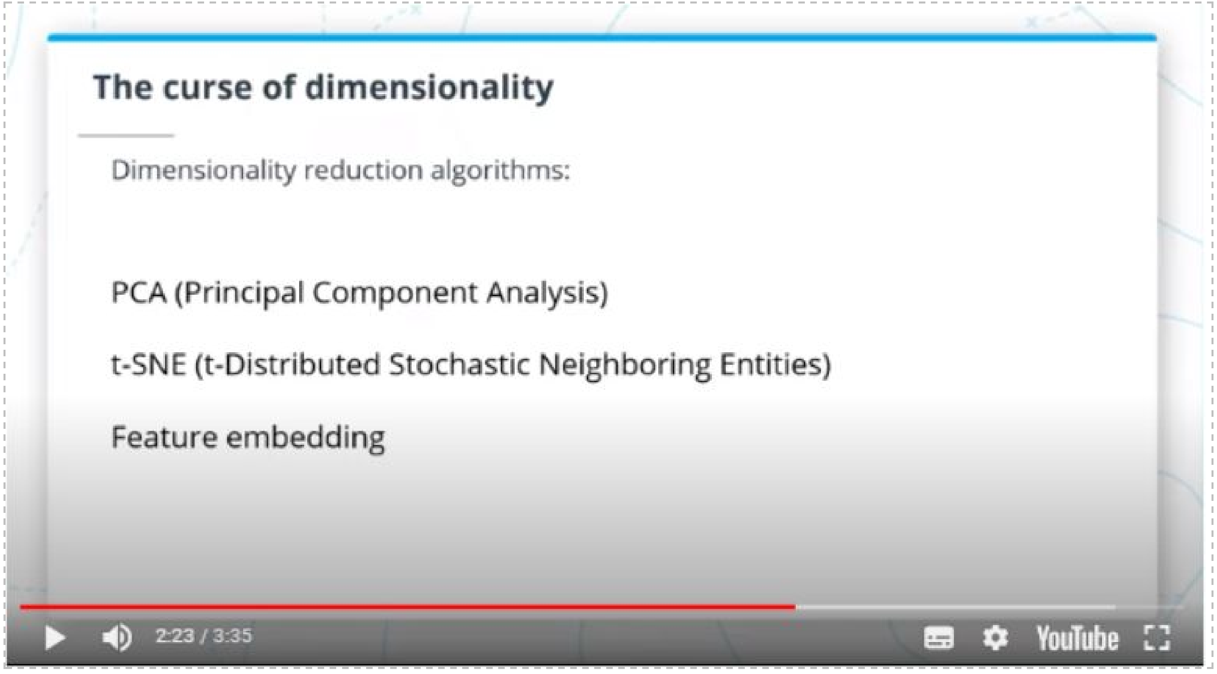

Feature Selection

Which are the features that are most useful for a given model?

Reasons for using Feature Selection:

Elimination of irrelevant, redundant, or (highly) correlated features

Reduction of dimensionality to increase performance



The curse of dimensionality

Dimensionality reduction algorithms:

PCA (Principal Component Analysis)

t-SNE (t-Distributed Stochastic Neighboring Entities)

Feature embedding

We can improve the situation of having too many features through **dimensionality reduction**.

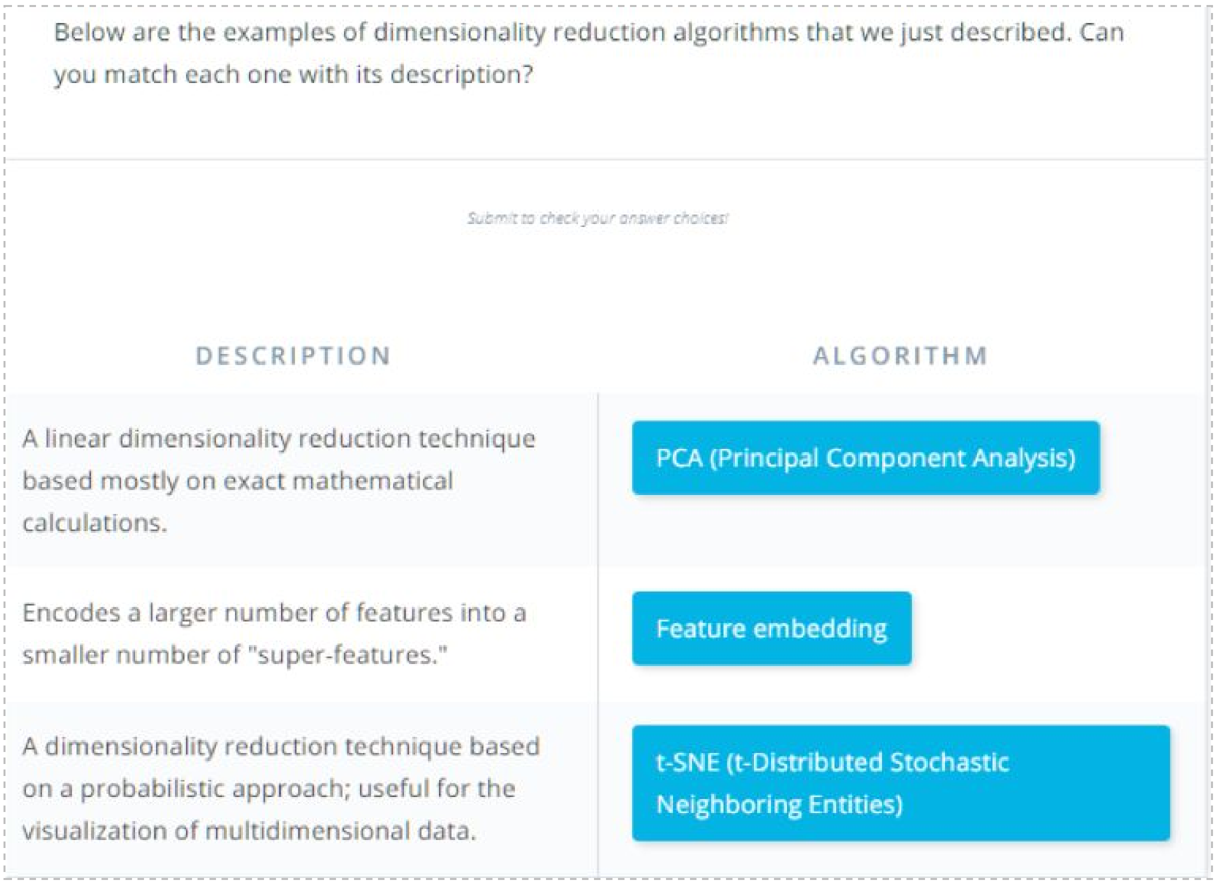
Commonly used techniques are:

- PCA (Principal Component Analysis)
- t-SNE (t-Distributed Stochastic Neighboring Entities)

- Feature embedding

Azure ML prebuilt modules:

- Filter-based feature selection: identify columns in the input dataset that have the greatest predictive power
- Permutation feature importance: determine the best features to use by computing the feature importance scores

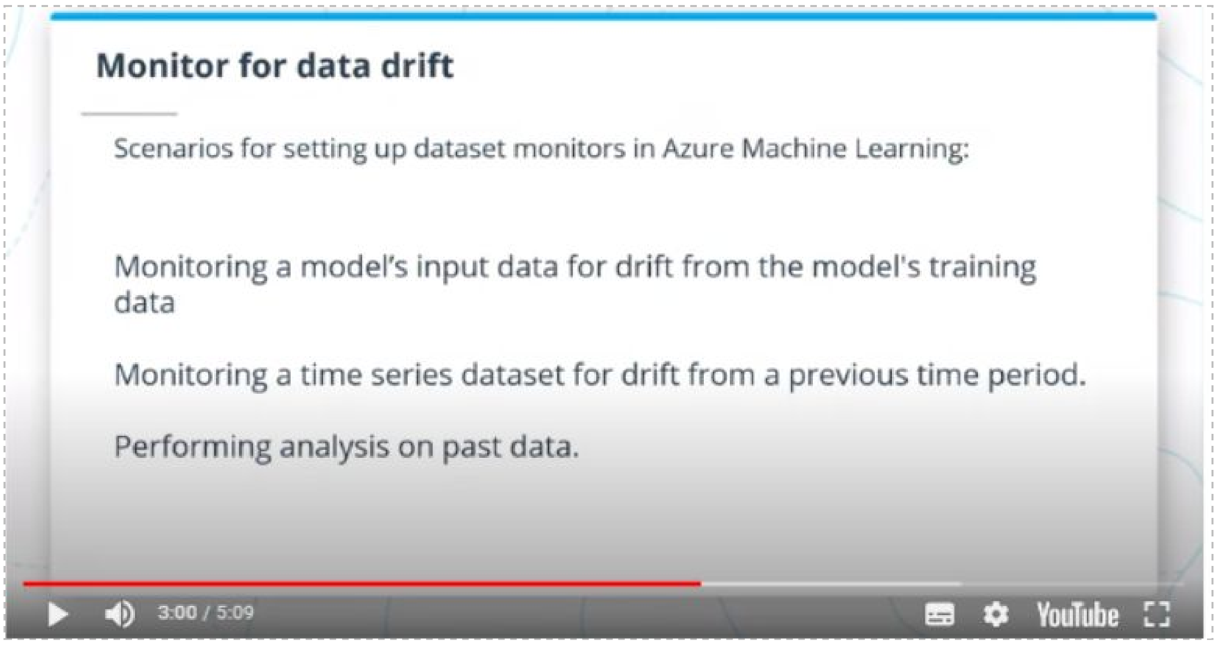


Below are the examples of dimensionality reduction algorithms that we just described. Can you match each one with its description?

Submit to check your answer choices!

DESCRIPTION	ALGORITHM
A linear dimensionality reduction technique based mostly on exact mathematical calculations.	PCA (Principal Component Analysis)
Encodes a larger number of features into a smaller number of "super-features."	Feature embedding
A dimensionality reduction technique based on a probabilistic approach; useful for the visualization of multidimensional data.	t-SNE (t-Distributed Stochastic Neighboring Entities)

Data Drift



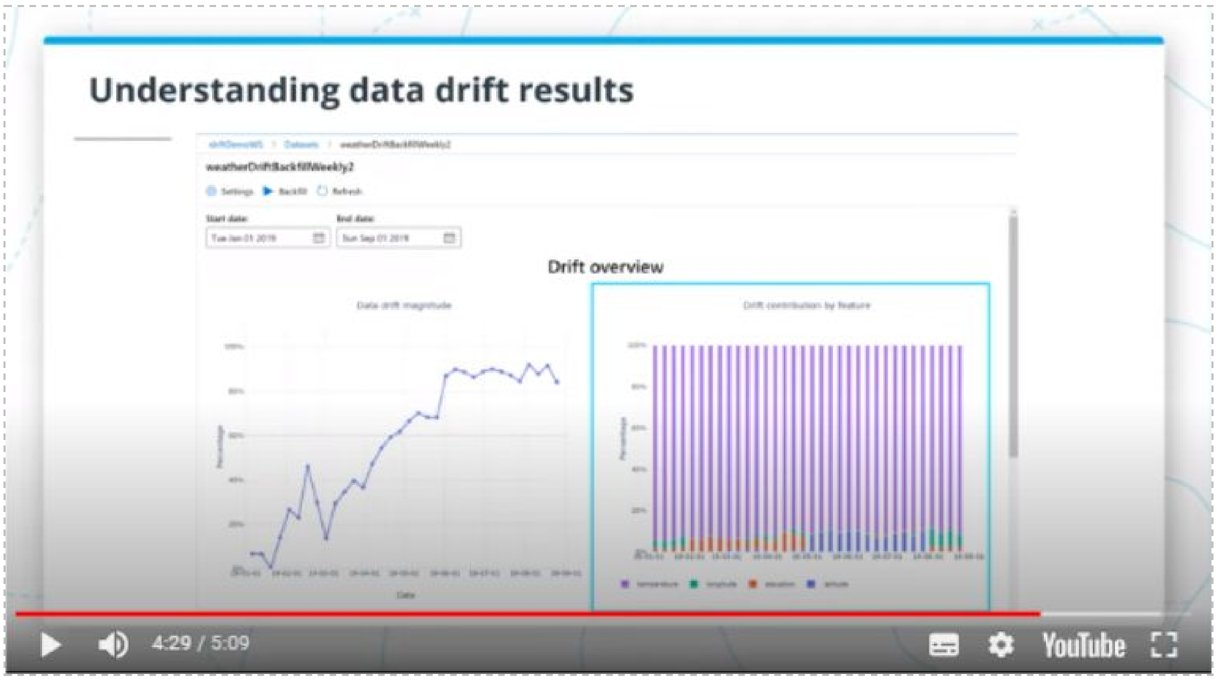
Monitor for data drift

Scenarios for setting up dataset monitors in Azure Machine Learning:

Monitoring a model's input data for drift from the model's training data

Monitoring a time series dataset for drift from a previous time period.

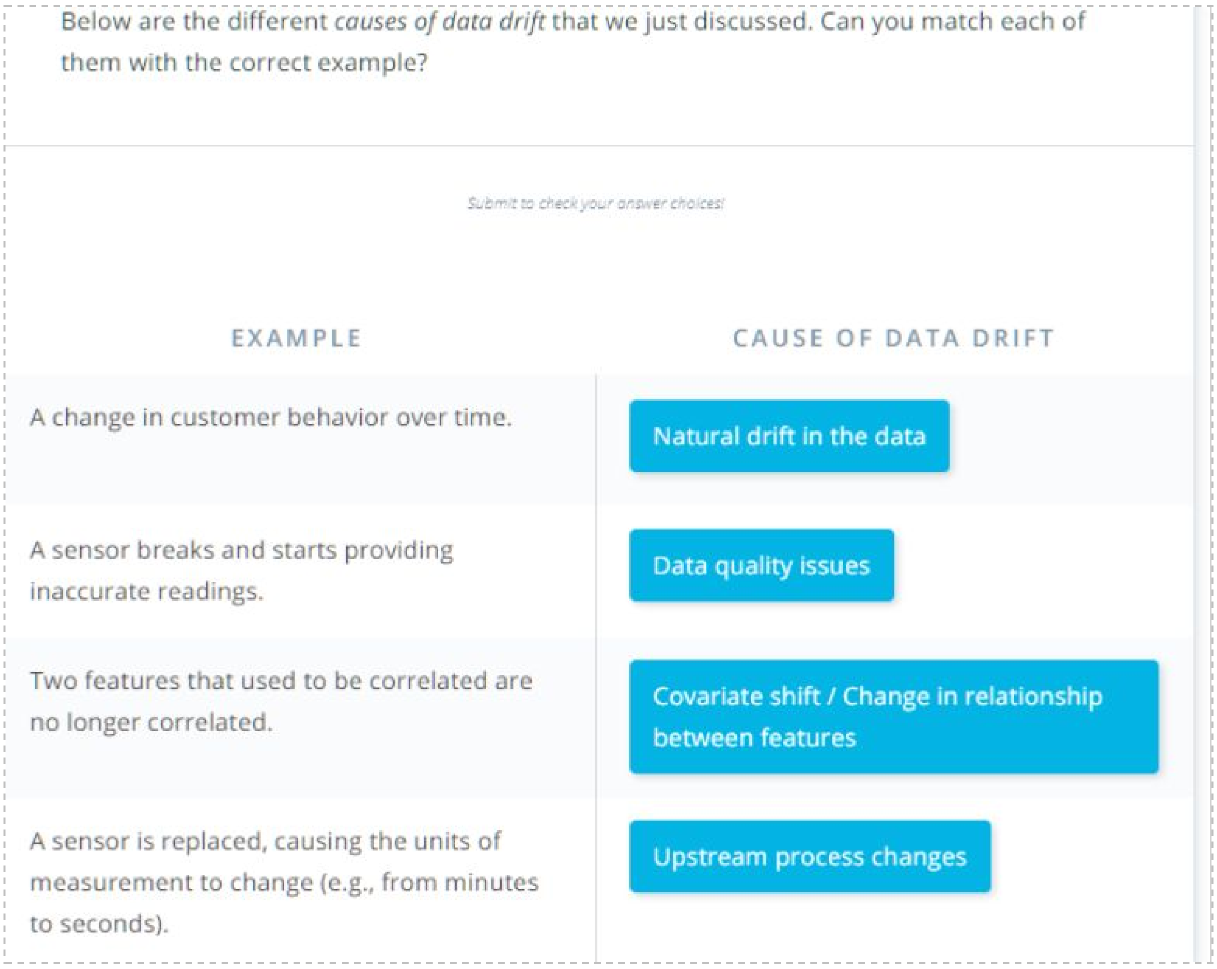
Performing analysis on past data.



Understanding data drift results

weatherDriftBackfillWeekly2

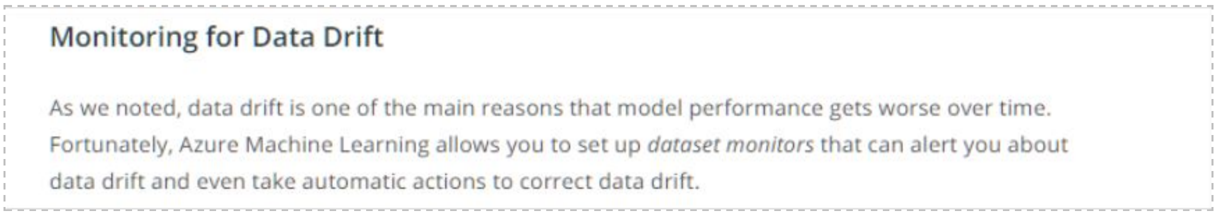
As we mentioned earlier, **data drift** is change in the input data for a model. Over time, data drift causes degradation in the model's performance, as the input data drifts farther and farther from the data on which the model was trained.



Below are the different *causes of data drift* that we just discussed. Can you match each of them with the correct example?

Submit to check your answer choices!

EXAMPLE	CAUSE OF DATA DRIFT
A change in customer behavior over time.	Natural drift in the data
A sensor breaks and starts providing inaccurate readings.	Data quality issues
Two features that used to be correlated are no longer correlated.	Covariate shift / Change in relationship between features
A sensor is replaced, causing the units of measurement to change (e.g., from minutes to seconds).	Upstream process changes



Monitoring for Data Drift

As we noted, data drift is one of the main reasons that model performance gets worse over time. Fortunately, Azure Machine Learning allows you to set up *dataset monitors* that can alert you about data drift and even take automatic actions to correct data drift.

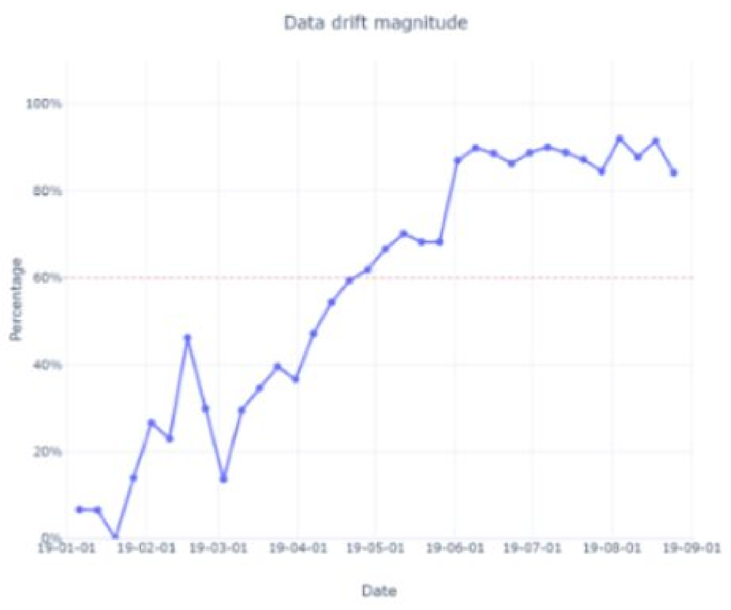
Remember, the process of monitoring for data drift involves:

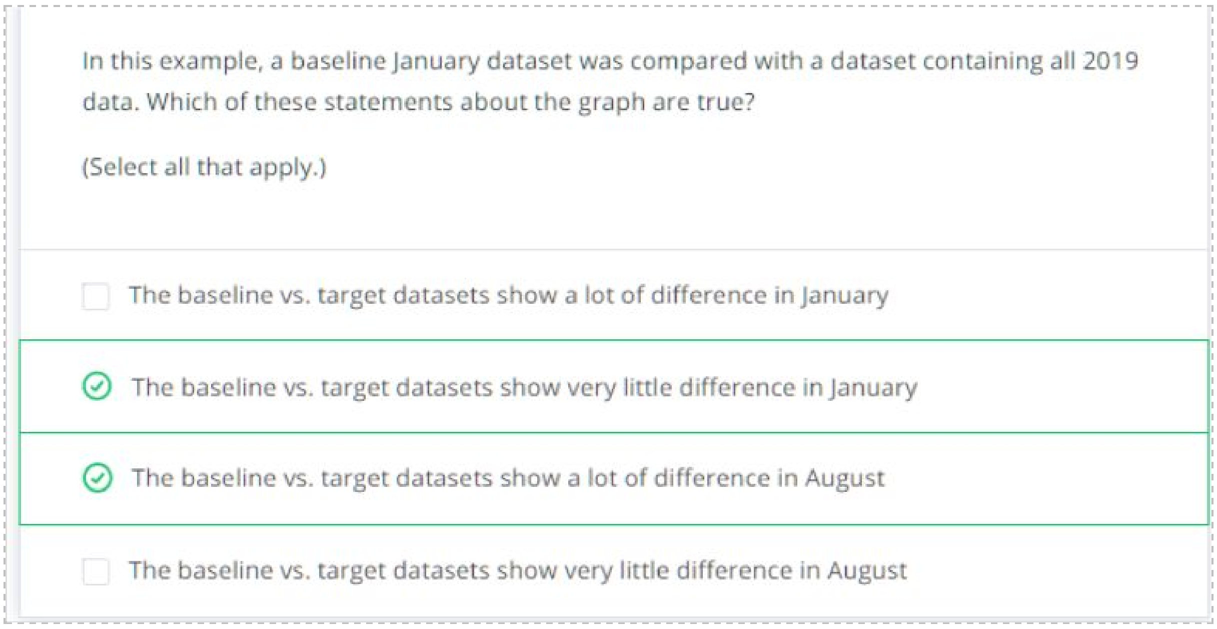
- Specifying a **baseline dataset** – usually the training dataset
- Specifying a **target dataset** – usually the input data for the model
- Comparing these two datasets over time, to monitor for differences

Here are a couple different types of comparisons you might want to make when monitoring for data drift:

- **Comparing input data vs. training data.** This is a proxy for model accuracy; that is, an increased difference between the input vs. training data is likely to result in a decrease in model accuracy.
- **Comparing different samples of time series data.** In this case, you are checking for a difference between one time period and another. For example, a model trained on data collected during one season may perform differently when given data from another time of year. Detecting this seasonal drift in the data will alert you to potential issues with your model's accuracy.

Here's the graph of data-drift magnitude that we looked at in the video:





In this example, a baseline January dataset was compared with a dataset containing all 2019 data. Which of these statements about the graph are true?

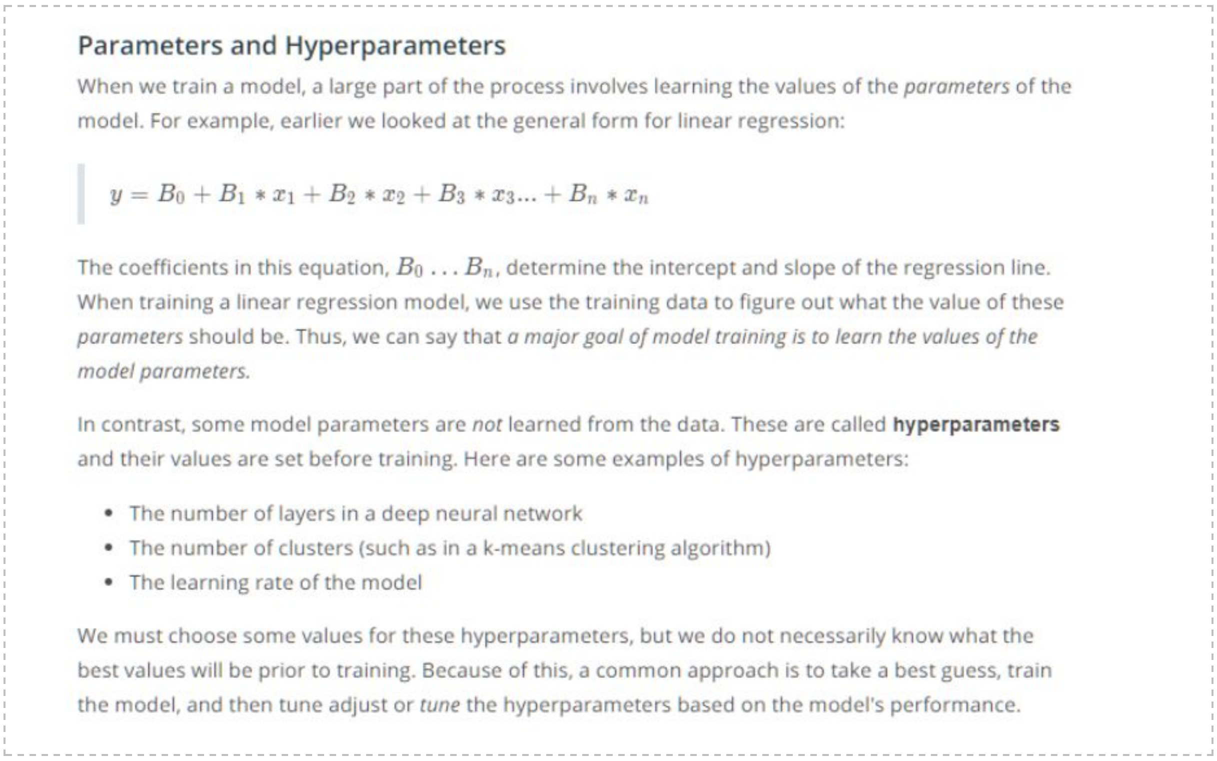
(Select all that apply.)

- ☐ The baseline vs. target datasets show a lot of difference in January
- ✓ The baseline vs. target datasets show very little difference in January
- ✓ The baseline vs. target datasets show a lot of difference in August
- ☐ The baseline vs. target datasets show very little difference in August

Model Training Basics

In this section, we will get into more detail on the steps involved in training a model, and then we'll get some hands-on practice with the process in the upcoming lab.

Remember that our ultimate goal is to produce a model we can use to make predictions. Put another way, we want to be able to give the model a set of input features, X, and have it predict the value of some output feature, y.



Parameters and Hyperparameters

When we train a model, a large part of the process involves learning the values of the *parameters* of the model. For example, earlier we looked at the general form for linear regression:

> $y = B_0 + B_1 * x_1 + B_2 * x_2 + B_3 * x_3 ... + B_n * x_n$

The coefficients in this equation, $B_0 \ldots B_n$, determine the intercept and slope of the regression line. When training a linear regression model, we use the training data to figure out what the value of these *parameters* should be. Thus, we can say that *a major goal of model training is to learn the values of the model parameters.*

In contrast, some model parameters are *not* learned from the data. These are called **hyperparameters** and their values are set before training. Here are some examples of hyperparameters:

- The number of layers in a deep neural network
- The number of clusters (such as in a k-means clustering algorithm)
- The learning rate of the model

We must choose some values for these hyperparameters, but we do not necessarily know what the best values will be prior to training. Because of this, a common approach is to take a best guess, train the model, and then tune adjust or *tune* the hyperparameters based on the model's performance.

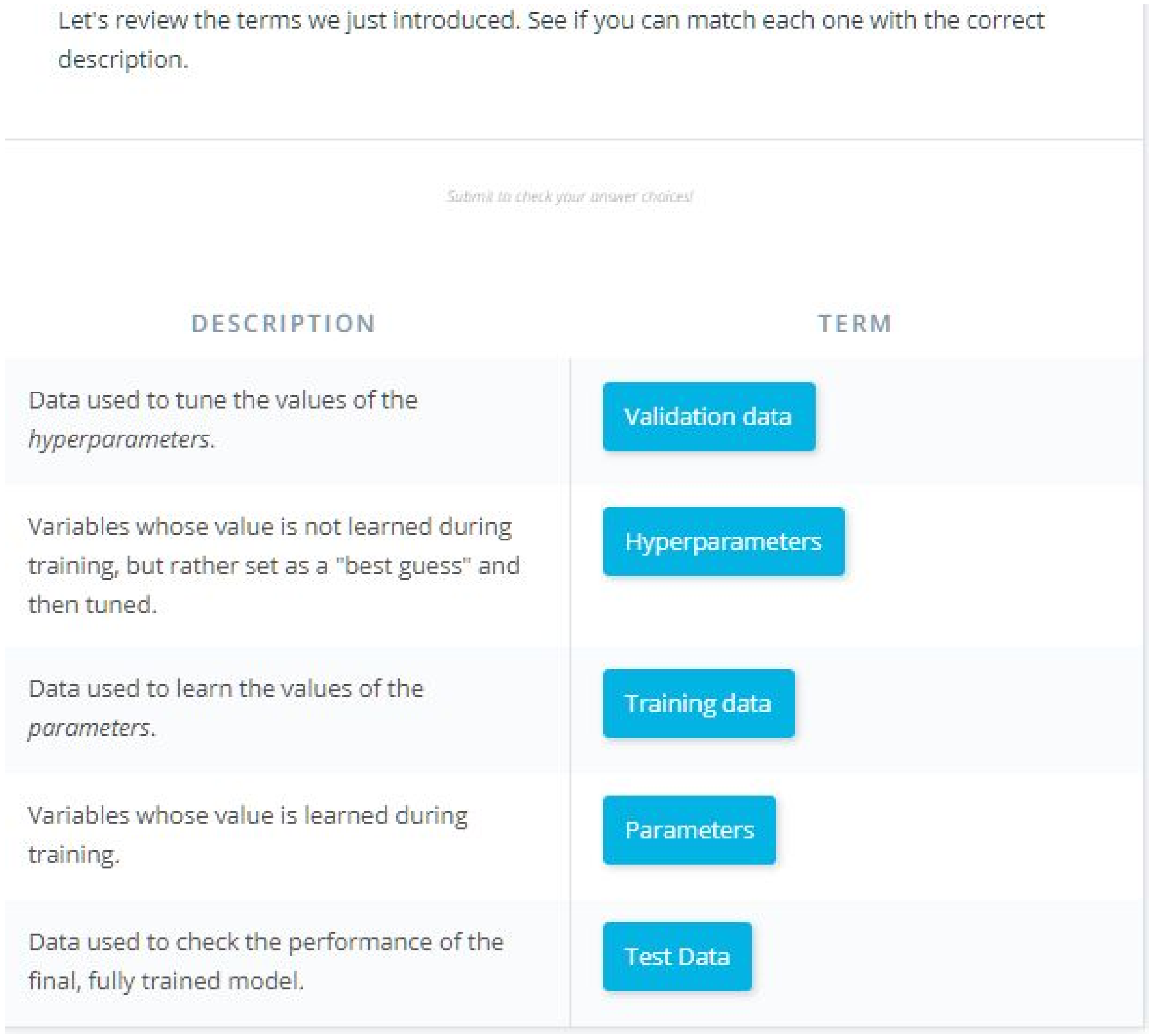
Splitting the Data

As mentioned in the video, we typically want to split our data into three parts:

- **Training data**
- **Validation data**
- **Test data**

We use the **training data** to learn the values for the *parameters*. Then, we check the model's performance on the **validation data** and *tune* the hyperparameters until the model performs well with the validation data. For instance, perhaps we need to have more or fewer layers in our neural network. We can adjust this hyperparameter and then test the model on the validation data once again to see if its performance has improved.

Finally, once we believe we have our finished model (with both parameters and hyperparameters optimized), we will want to do a final check of its performance—and we need to do this on some fresh **test data** that we did not use during the training process.

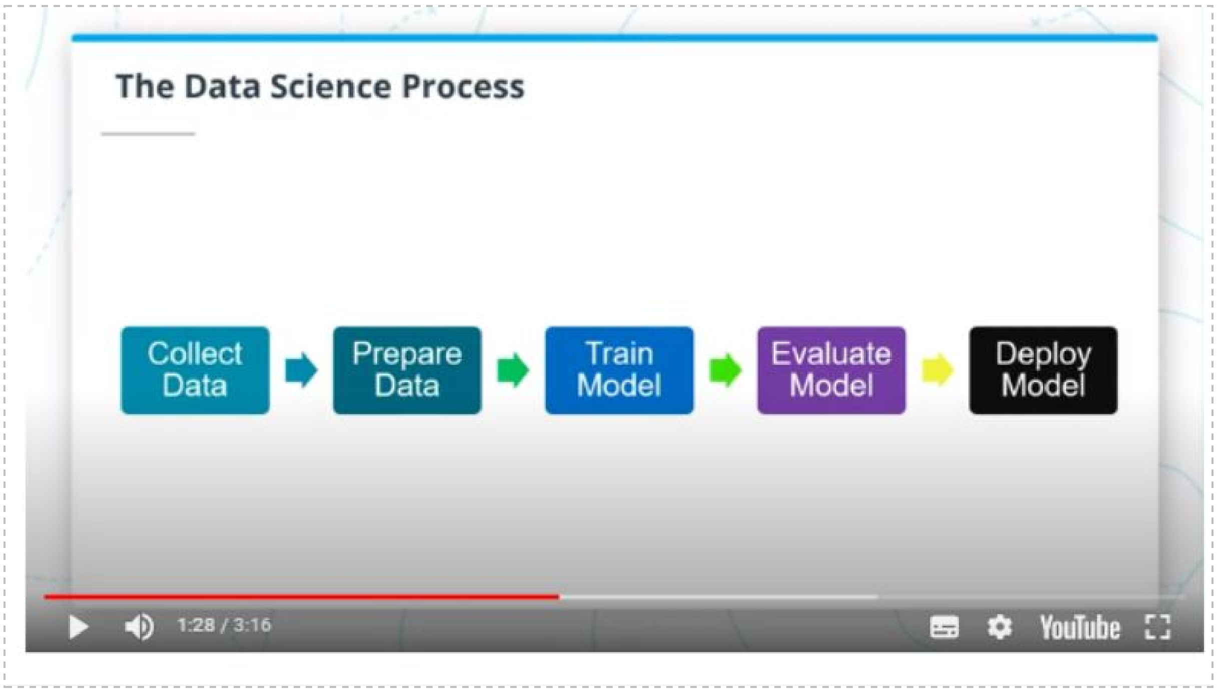


Let's review the terms we just introduced. See if you can match each one with the correct description.

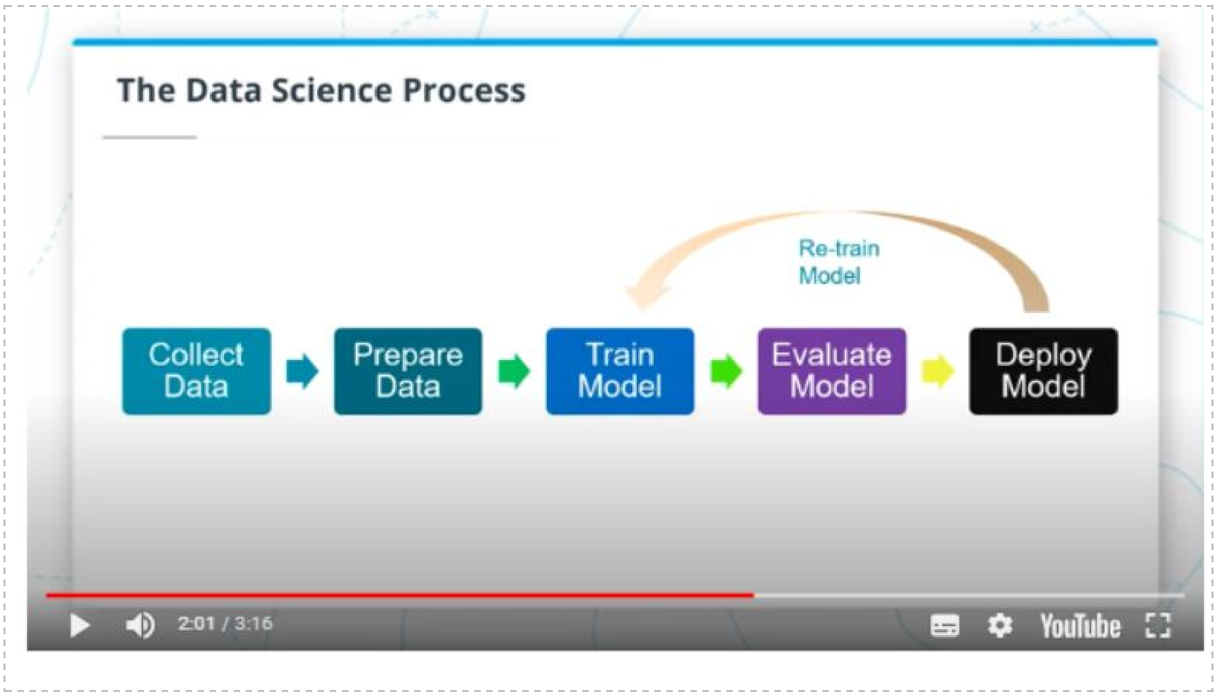
Submit to check your answer choices!

DESCRIPTION	TERM
Data used to tune the values of the *hyperparameters*.	Validation data
Variables whose value is not learned during training, but rather set as a "best guess" and then tuned.	Hyperparameters
Data used to learn the values of the *parameters*.	Training data
Variables whose value is learned during training.	Parameters
Data used to check the performance of the final, fully trained model.	Test Data

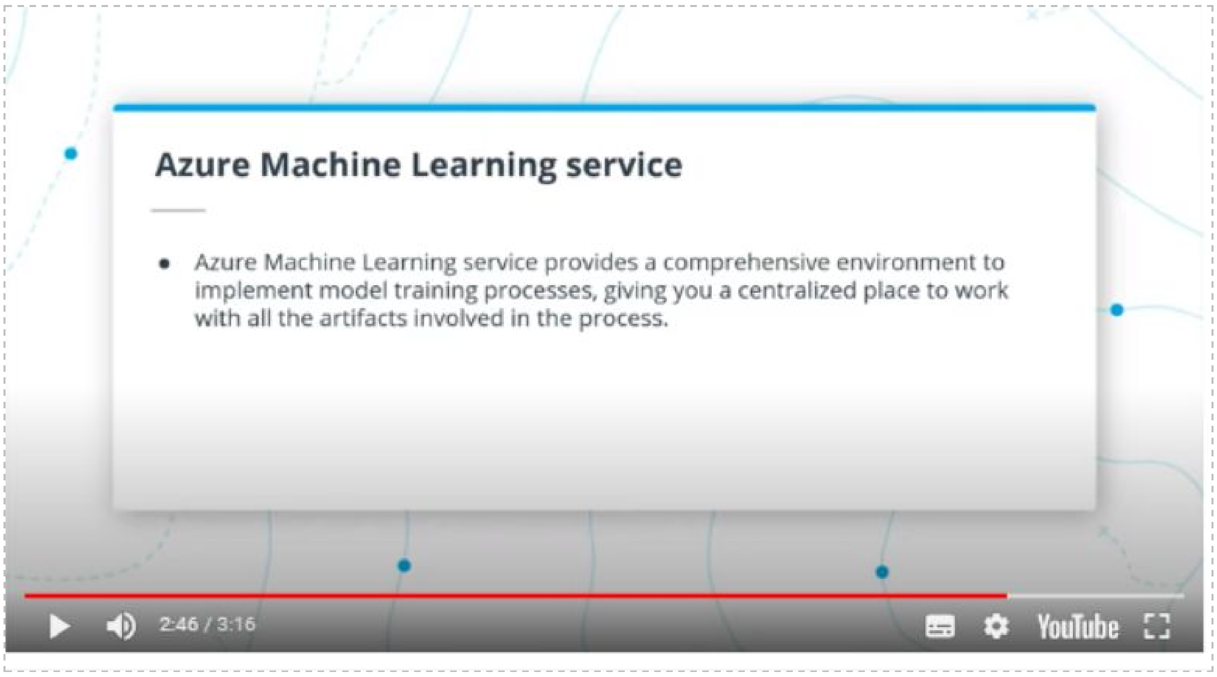
Model Training in Azure Machine Learning



The Data Science Process



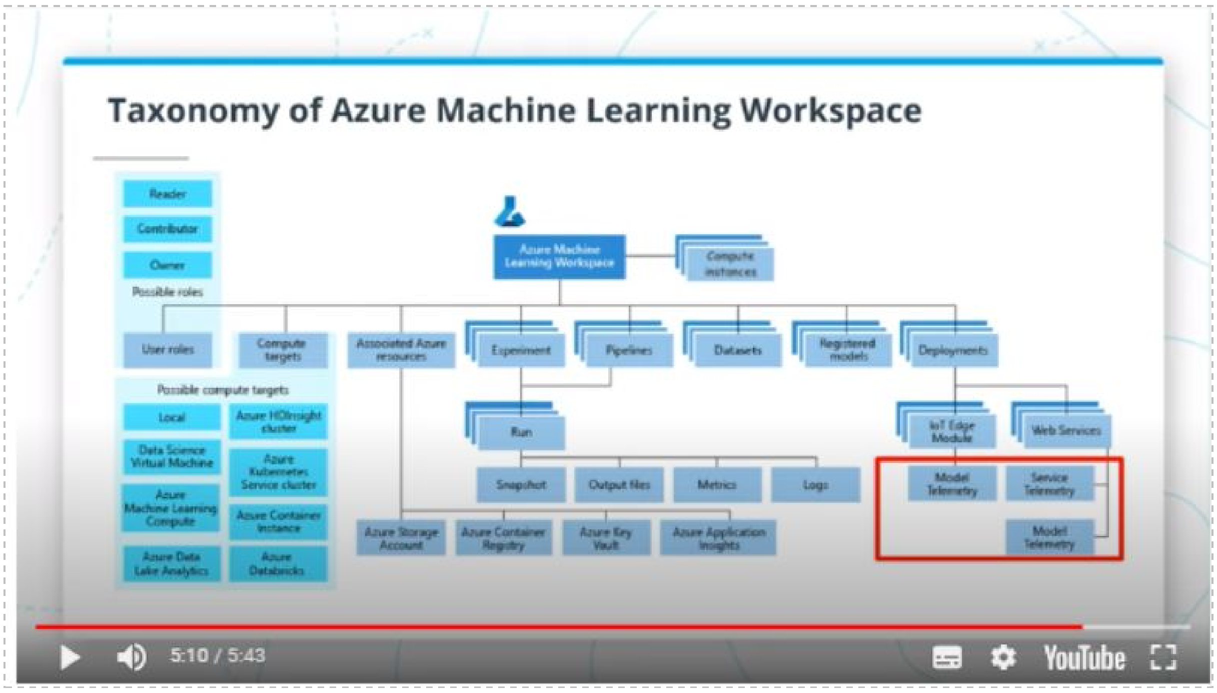
The Data Science Process



Azure Machine Learning service

- Azure Machine Learning service provides a comprehensive environment to implement model training processes, giving you a centralized place to work with all the artifacts involved in the process.

Taxonomy of Azure Machine Learning



Taxonomy of Azure Machine Learning Workspace

Below are some of the components of Azure Machine Learning. Can you match each of them with the correct description?

(We'll get practice working with all of these in the labs, but it will help if you get some general familiarity with them now.)

Run Model telemetry

DESCRIPTION	COMPONENT
The centralized place for working with all the components of the machine learning process.	Workspace
A container that helps you organize the model training process.	Experiment
A service that provides snapshots and versioning for your trained models.	Model registry
A cloud-based workstation that gives you access to various development environments, such as Jupyter Notebooks.	Compute instance