What is Machine Learning?

One of our goals in this lesson is to help you get a clearer, more specific understanding of what machine learning is and how it differs from other approaches.w

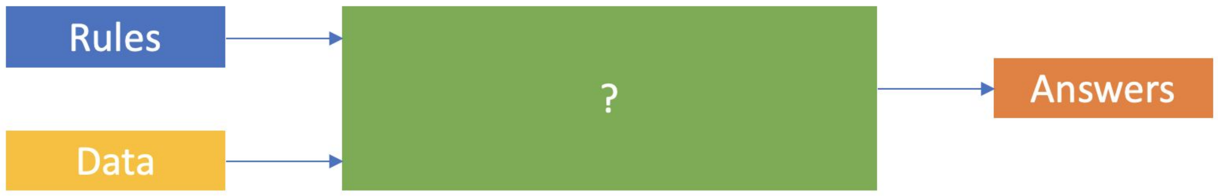
Let's start with a classic definition. If you look up the term in a search engine, you might find something like this:

***Machine learning** is a data science technique used to extract patterns from data, allowing computers to identify related data, and forecast future outcomes, behaviors, and trends.*

Let's break that down a little. One important component of machine learning is that we are taking some *data* and using it to *make predictions* or *identify important relationships*. But looking for patterns in data is done in traditional data science as well. So how does machine learning differ? In this next video, we'll go over a few examples to illustrate the difference between machine learning and traditional programming.

QUESTION 1 OF 5

What type of approach is shown in this image?

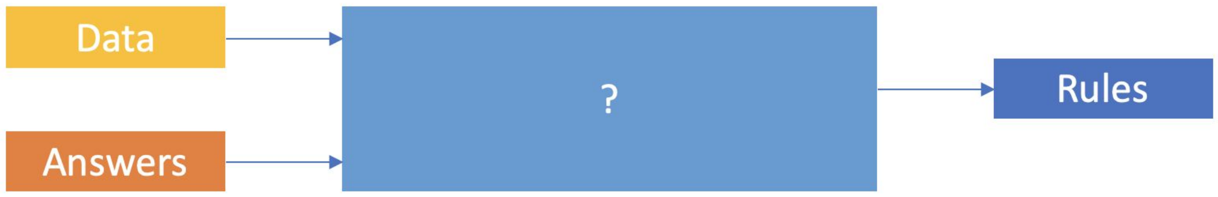


- Traditional Programming
- Machine Learning

SUBMIT

QUESTION 2 OF 5

What type of approach is shown in this image?



- Traditional Programming
- Machine Learning

SUBMIT

QUESTION 3 OF 5

Imagine you want to create a function that multiplies two numbers together (e.g., given the inputs `2` and `3`, the function will generate the output `6`).

What approach is best suited to this problem?

- **Traditional Programming**
- Machine Learning

SUBMIT

QUESTION 4 OF 5

Now imagine that you have some images that contain handwritten numbers. You want to create a program that will recognize which number is in each picture, but you're not sure exactly what characteristics can be used to best tell the numbers apart.

Which is the best approach for creating this program?

- Traditional programming
- **Machine learning**

SUBMIT

QUESTION 5 OF 5

In *traditional programming*, the inputs of hard-coded rules and data are used to arrive at the output of answers, but in *machine learning* the approach is quite different.

Mark all of the options below that are true statements about **machine learning**.

- **Data is input to train an algorithm**
- **Historical answers are input to train an algorithm**
- Rules are explicitly programmed
- **Rules are the output learned by the algorithm**

SUBMIT

NEXT

Applications of Machine Learning

The applications of machine learning are extremely broad! And the opportunities cut across industry verticals. Whether the industry is healthcare, finance, manufacturing, retail, government, or education, there is enormous potential to apply machine learning to solve problems in more efficient and impactful ways.

We'll take a tour through some of the major areas where machine learning is applied, mainly just to give you an idea of the scope and type of problems that machine learning is commonly used to address.

Examples of Applied Machine Learning

Machine learning is used to solve an extremely diverse range of problems. For your reference, here are all the examples we discussed in the video, along with links to further reading in case you are curious and want to learn more about any of them:

Automate the recognition of disease

Trained physicians can only review and evaluate a limited volume of patients or patient images (X-rays, sonograms, etc.). Machine learning can be used to spot the disease, hence reducing physician burnout. For example, Google has trained a deep learning model to detect breast cancer and Stanford researchers have used deep learning models to diagnose skin cancer.

Recommend next best actions for individual care plans

With the mass digitization of patient data via systems that use EMRs (Electronic Medical Records) and EHRs (Electronic Health Records), machine learning can be used to help build effective individual care plans. For example, IBM Watson Oncology can help clinicians explore potential treatment options. More examples of how machine learning impacts healthcare can be found here.

Enable personalized, real-time banking experiences with chatbots

You've likely encountered this when you call a customer service number. Machine learning can be used to intercept and handle common, straightforward issues through chat and messaging services, so customers can quickly and independently resolve simple issues that would otherwise have required human intervention. With the chatbot, a customer can simply type in a question and the bot engages to surface the answer. Refer to this article to find more information about chatbot powered machine learning.

Identify the next best action for the customer

Real-time insights that incorporate machine learning tools—such as sentiment analysis—can help organizations assess the likelihood of a deal closing or the level of a customer's loyalty. Personally-tailored recommendations powered by machine learning can engage and delight customers with information and offers that are relevant to them.

Capture, prioritize, and route service requests to the correct employee, and improve response times

A busy government organization gets innumerable service requests on an annual basis. Machine learning tools can help to capture incoming service requests, to route them to the correct employee in real-time, to refine prioritization, and improve response times. Can check out this article if you're curious to learn more about ticket routing

NEXT

Brief History of Machine Learning

QUIZ QUESTION

There is often confusion between the terms *machine learning*, *deep learning*, and *artificial intelligence*. See if you can match each term with its description:

Submit to check your answer choices!

DESCRIPTION	**TERM**
A broad term that refers to computers thinking more like humans.	Artificial intelligence

A subcategory of artificial intelligence that involves learning from data without being explicitly programmed.

Machine learning

A subcategory of machine learning that uses a layered neural-network architecture originally inspired by the human brain.

Deep learning

SUBMIT

Further Reading

- *What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?* by Michael Copeland at NVIDIA

NEXT

The Data Science Process

Big data has become part of the lexicon of organizations worldwide, as more and more organizations look to leverage data to drive informed business decisions. With this evolution in business decision-making, the amount of raw data collected, along with the number and diversity of data sources, is growing at an astounding rate. This data presents enormous potential.

Raw data, however, is often noisy and unreliable and may contain missing values and outliers. Using such data for modeling can produce misleading results. For the data scientist, the ability to combine large, disparate data sets into a format more appropriate for analysis is an increasingly crucial skill.

The data science process typically starts with collecting and preparing the data before moving on to training, evaluating, and deploying a model. Let's have a look.

QUESTION 1 OF 3

Here are the typical steps of the data science process that we just discussed. Can you remember the correct order?

Submit to check your answer choices!

STEP	DESCRIPTION
Steps 1 & 2	Collect and prepare the data
Steps 3 & 4	Train the model and evaluate its performance
Steps 5 & 6	Deploy the model and then retrain as necessary

SUBMIT

QUESTION 2 OF 3

Here are some of the steps once again, along with some of the actions that you would carry out during those steps. Can you match the step with the appropriate action?

(Again, first try to do it from memory—but have a look at the text or video above if you get stuck.)

Submit to check your answer choices!

ACTION	**WHICH STEP OF THE PROCESS?**
Package the model and dependencies	Deploy the model
Run the model through a final exam using data from your validation data set	Evaluate the model
Create features needed for the model	Prepare the data
Select the algorithm, and prepare training, testing, and validation data sets	

Train the model

SUBMIT

QUESTION 3 OF 3

In machine learning, often you have to tune parameters for the chosen learning algorithm to improve the performance on relevant metrics, such as prediction accuracy. At what stage of the data science lifecycle do you optimize the parameters?

- Training the model
- Evaluating the model
- Deploying the model

SUBMIT

NEXT

Lesson 2:

Introduction to Machine Learning

SEARCH
RESOURCES
CONCEPTS

Common Types of Data

SEND FEEDBACK

Common Types of Data

It's All Numerical in the End

Note that although we've described *numerical data* as a distinct category, it is actually involved in some way with all of the data types we've described. With the example of stock performance (above) the stock prices are numerical data points. So why do we give this as an example of "time-series data" rather than "numerical data"? It is the ordering of the numerical data points *across points in time* that leads us to call the data *time-series data*.

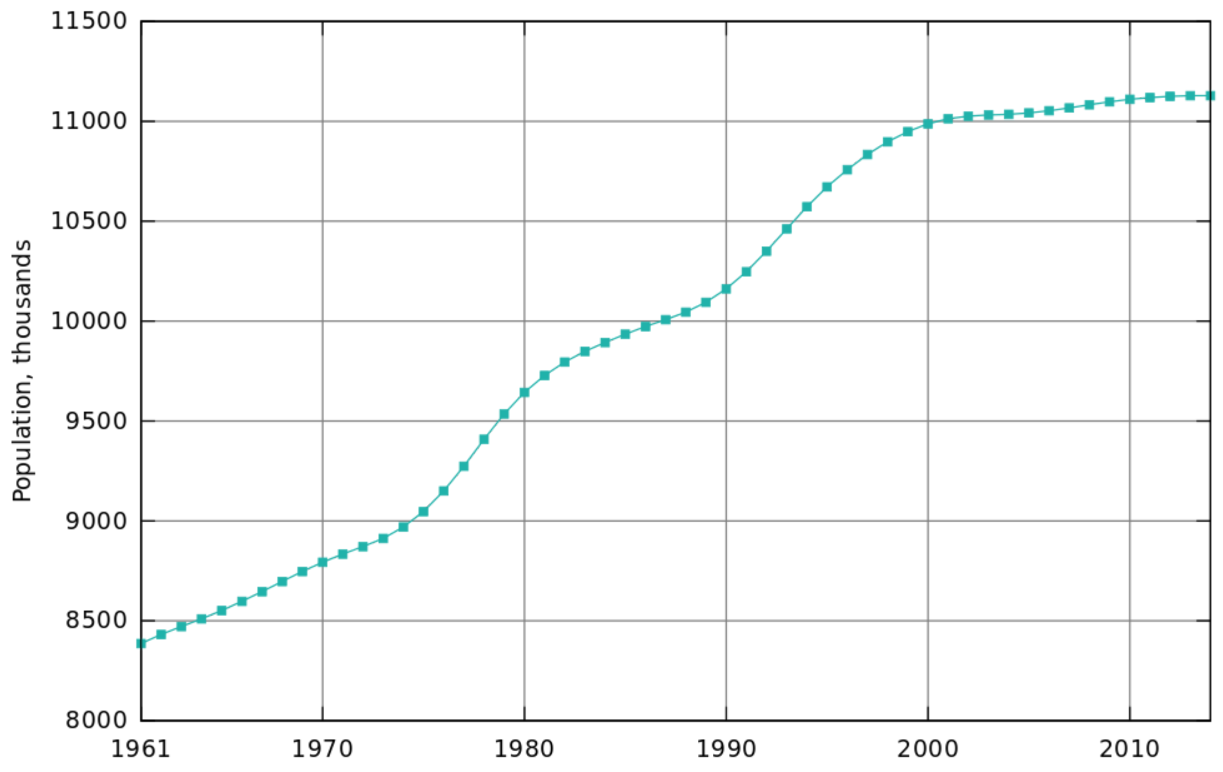
What is more, all data in machine learning eventually ends up being numerical, regardless of whether it is numerical in its original form, so it can be processed by machine learning algorithms.

For example, we may want to use gender information in the dataset to predict if an individual has heart disease. Before we can use this information with a machine learning algorithm, we need to transfer male vs. female into numbers, for instance, 1 means a person is male and 2 means a person is female, so it can be processed. Note here that the value 1 or 2 does not carry any meaning.

Another example would be using pictures uploaded by customers to identify if they are satisfied with the service. Pictures are not initially in numerical form but they will need to be transformed into RGB values, a set of numerical values ranging from 0 to 255, to be processed.

QUESTION 1 OF 2

Have a look at this graph:



Population of Greece since 1961 (*Wikimedia Commons*)

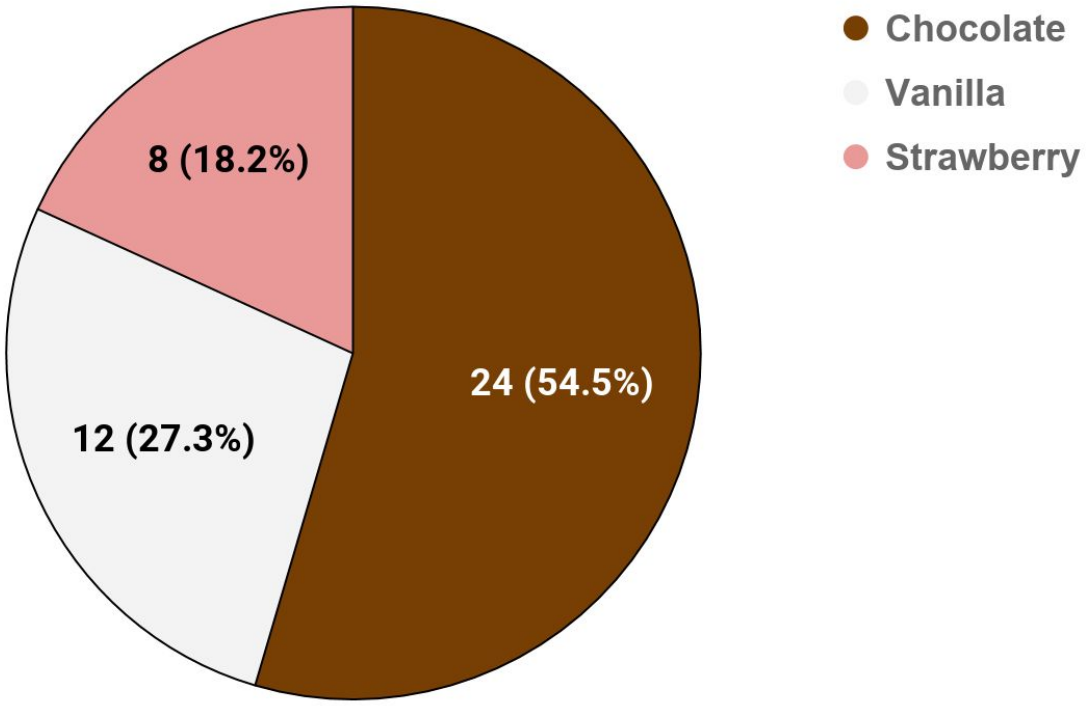
What type of data is this?

- Numerical
- Time-Series
- Categorical
- Text

SUBMIT

QUESTION 2 OF 2

Have a look at this chart showing the number of people who like each flavor of ice cream:



What type of data is this?

- Numerical
- Time-Series
- **Categorical**
- Text

SUBMIT

NEXT

Lesson 2:

Introduction to Machine Learning

SEARCH
RESOURCES
CONCEPTS

Tabular Data

SEND FEEDBACK

Tabular Data

In machine learning, the most common type of data you'll encounter is **tabular data**—that is, data that is arranged in a data *table*. This is essentially the same format as you work with when you look at data in a spreadsheet.

Here's an example of tabular data showing some different clothing products and their properties:

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

Notice how tabular data is arranged in **rows** and **columns**.

QUESTION 1 OF 2

Looking at the table above, can you figure out what the **rows** vs. **columns** are for?

- **Each row describes a single product (e.g., a shirt), while each column describes a property the products can have (e.g., the color of the product)**
- Each column describes a single product (e.g., a shirt), while each row describes a property the products can have (e.g., the color of the product)

SUBMIT

QUESTION 2 OF 2

Below are the components of a table. What does each of these components represent?

Submit to check your answer choices!

WHAT IT REPRESENTS	COMPONENT
An item or entity.	Row
A property that the items or entities in the table can have.	Column
A single value.	Cell

SUBMIT

Vectors

It is important to know that in machine learning we ultimately always work with numbers or specifically *vectors*.

*A **vector** is simply an array of numbers, such as `(1, 2, 3)`—or a nested array that contains other arrays of numbers, such as `(1, 2, (1, 2, 3))`.*

Vectors are used heavily in machine learning. If you have taken a basic course in linear algebra, then you are probably in good shape to begin learning about how they are used in machine learning. But if linear algebra and vectors are totally new to you, there are some great free resources available to help you learn. You may want to have a look at Khan Academy's excellent introduction to the topic here or check out Udacity's free Linear Algebra Refresher Course.

For now, the main points you need to be aware of are that:

- All non-numerical data types (such as images, text, and categories) must eventually be represented as numbers
- In machine learning, the numerical representation will be in the form of an *array of numbers*—that is, a *vector*

As we go through this course, we'll look at some different ways to take non-numerical data and *vectorize* it (that is, transform it into vector form).

NEXT

Scaling Data

SEND FEEDBACK

Scaling Data

Scaling data means transforming it so that the values fit within some range or scale, such as 0–100 or 0–1. There are a number of reasons why it is a good idea to scale your data before feeding it into a machine learning algorithm.

Let's consider an example. Imagine you have an image represented as a set of RGB values ranging from 0 to 255. We can scale the range of the values from 0–255 down to a range of 0–1. This scaling process will not affect the algorithm output since every value is scaled in the same way. But it can speed up the training process, because now the algorithm only needs to handle numbers less than or equal to 1.

Two common approaches to scaling data include **standardization** and **normalization**.

Standardization

Standardization *rescales data so that it has a mean of 0 and a standard deviation of 1.*

The formula for this is:

$(x - \mu)/\sigma$

We subtract the mean (μ) from each value (x) and then divide by the standard deviation (σ). To understand why this works, it helps to look at an example.

Suppose that we have a sample that contains three data points with the following values:

```
50
100
150
```

The mean of our data would be `100`, while the sample standard deviation would be `50`.

Let's try standardizing each of these data points. The calculations are:

```
(50 - 100)/50 = -50/50 = -1
(100 - 100)/50 = 0/50 = 0
(150 - 100)/50 = 50/50 = 1
```

Thus, our transformed data points are:

```
-1
0
1
```

Again, the result of the standardization is that our data distribution now has a mean of 0 and a standard deviation of 1.

Normalization

***Normalization** rescales the data into the range [0, 1].*

The formula for this is:

$(x - xmin)/(xmax - xmin)$

For each individual value, you subtract the minimum value ($xmin$) for that input in the training dataset, and then divide by the range of the values in the training dataset. The range of the values is the difference between the maximum value ($xmax$) and the minimum value ($xmin$).

Let's try working through an example with those same three data points:

```
50
100
150
```

The minimum value ($xmin$) is `50`, while the maximum value ($xmax$) is `150`. The range of the values is $xmax - xmin$ = 150 − 50 = 100.

Plugging everything into the formula, we get:

```
(50 - 50)/100 = 0/100 = 0
(100 - 50)/100 = 50/100 = 0.5
(150 - 50)/100 = 100/100 = 1
```

Thus, our transformed data points are:

```
0
0.5
1
```

Again, the goal was to rescale our data into values ranging from 0 to 1—and as you can see, that's exactly what the formula did.

QUESTION 1 OF 3

Which of the below refers to **standardization** and which refers to **normalization**?

Submit to check your answer choices!

DESCRIPTION

STANDARDIZATION OR NORMALIZATION?

DESCRIPTION	STANDARDIZATION OR NORMALIZATION?
Rescales the data to have mean = 0 and standard deviation = 1	Standardization
Rescales the data into the range [0, 1]	Normalization
$(x - xmin)/(xmax - xmin)$	Normalization
$(x - \mu)/\sigma$	Standardization

SUBMIT

QUESTION 2 OF 3

Standardize -5,10,15. Knowing that the mean is 7 and the standard deviation is 10

- **-1.2, 0.3, 0.8**
- -1, 0.5, 0.8
- -1.2, 0.3, 0.5
- -1.2, 0.5, 0.8

SUBMIT

QUESTION 3 OF 3

Normalize -5,10,15. Knowing that the mean is 7 and the standard deviation is 10.

- -0.5, 0.65, 1.0
- **0.0, 0.75, 1.0**
- 0.5, 0.75, 0.8
- 0.0, 0.65, 0.8

SUBMIT

NEXT

Encoding Categorical Data

As we've mentioned a few times now, machine learning algorithms need to have data in numerical form. Thus, when we have *categorical* data, we need to encode it in some way so that it is represented numerically.

There are two common approaches for encoding categorical data: **ordinal encoding** and **one hot encoding**.

Ordinal Encoding

In **ordinal encoding**, we simply convert the categorical data into integer codes ranging from `0` to `(number of categories - 1)`. Let's look again at our example table of clothing products:

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

If we apply ordinal encoding to the `Make` property, we get the following:

Make	**Encoding**
A&F	0

Guess	1
Tillys	2

And if we apply it to the Color property, we get:

Color	**Encoding**
Red	0
Green	1
Blue	2

Using the above encoding, the transformed table is shown below:

SKU	Make	Color	Quantity	Price
908721	1	2	789	45.33
456552	2	0	244	22.91
789921	0	1	387	25.92
872266	1	2	154	17.56

One of the potential drawbacks to this approach is that it implicitly assumes an order across the categories. In the above example, `Blue` (which is encoded with a value of `2`) seems to be *more* than `Red` (which is encoded with a value of `1`), even though this is in fact not a meaningful way of comparing those values. This is not *necessarily* a problem, but it is a reason to be cautious in terms of how the encoded data is used.

One-Hot Encoding

One-hot encoding is a very different approach. In one-hot encoding, we transform each categorical value into a column. If there are `n` categorical values, `n` new columns are added. For example, the `Color` property has three categorical values: `Red`, `Green`, and `Blue`, so three new columns `Red`, `Green`, and `Blue` are added.

If an item belongs to a category, the column representing that category gets the value `1`, and all other columns get the value `0`. For example, item 908721 (first row in the table) has the color blue, so we put `1` into that `Blue` column for 908721 and `0` into the `Red` and `Green` columns. Item 456552 (second row in the table) has color red, so we put `1` into that `Red` column for 456552 and `0` into the `Green` and `Blue` columns.

If we do the same thing for the `Make` property, our table can be transformed as follows:

SKU	A&F	Guess	Tillys	Red	Green	Blue	Quantity	Price
908721	0	1	0	0	0	1	789	45.33
456552	0	0	1	1	0	0	244	22.91
789921	1	0	0	0	1	0	387	25.92
872266	0	1	0	0	0	1	154	17.56

One drawback of one-hot encoding is that it can potentially generate a very large number of columns.

QUESTION 1 OF 4

Have a look at this tabular data:

ID	Mammal	Reptile	Fish
012	1	0	0
204	0	0	1
009	0	1	0

105	1	0	0

What type of encoding has been performed on this?

- Ordinal encoding
- One-hot encoding

SUBMIT

QUESTION 2 OF 4

Looking again at the table in the previous question, what category is animal `204`?

- Mammal
- Reptile
- **Fish**

SUBMIT

QUESTION 3 OF 4

Again looking at the above animals table, suppose we do the following:

1. Add two new categories, `Amphibian` and `Bird`
2. Add one bird with ID `303` in the table

Which one of the following statements is correct about the new table?

- There are 5 columns in the new table including the `ID` column
- Animal `303` has `1` in the `Mammal` column
- **The `Amphibian` column has `0` for all animals**
- Animal `303` has `0` in the `Bird` column

SUBMIT

QUESTION 4 OF 4

John is looking to train his first machine learning model. One of his inputs includes the size of the T-Shirts, with possible values of XS, S, M, L, and XL. What is the best approach John can employ to preprocess the T-Shirt size input feature?

- Standardization
- Normalization
- **One Hot Encoding**

SUBMIT

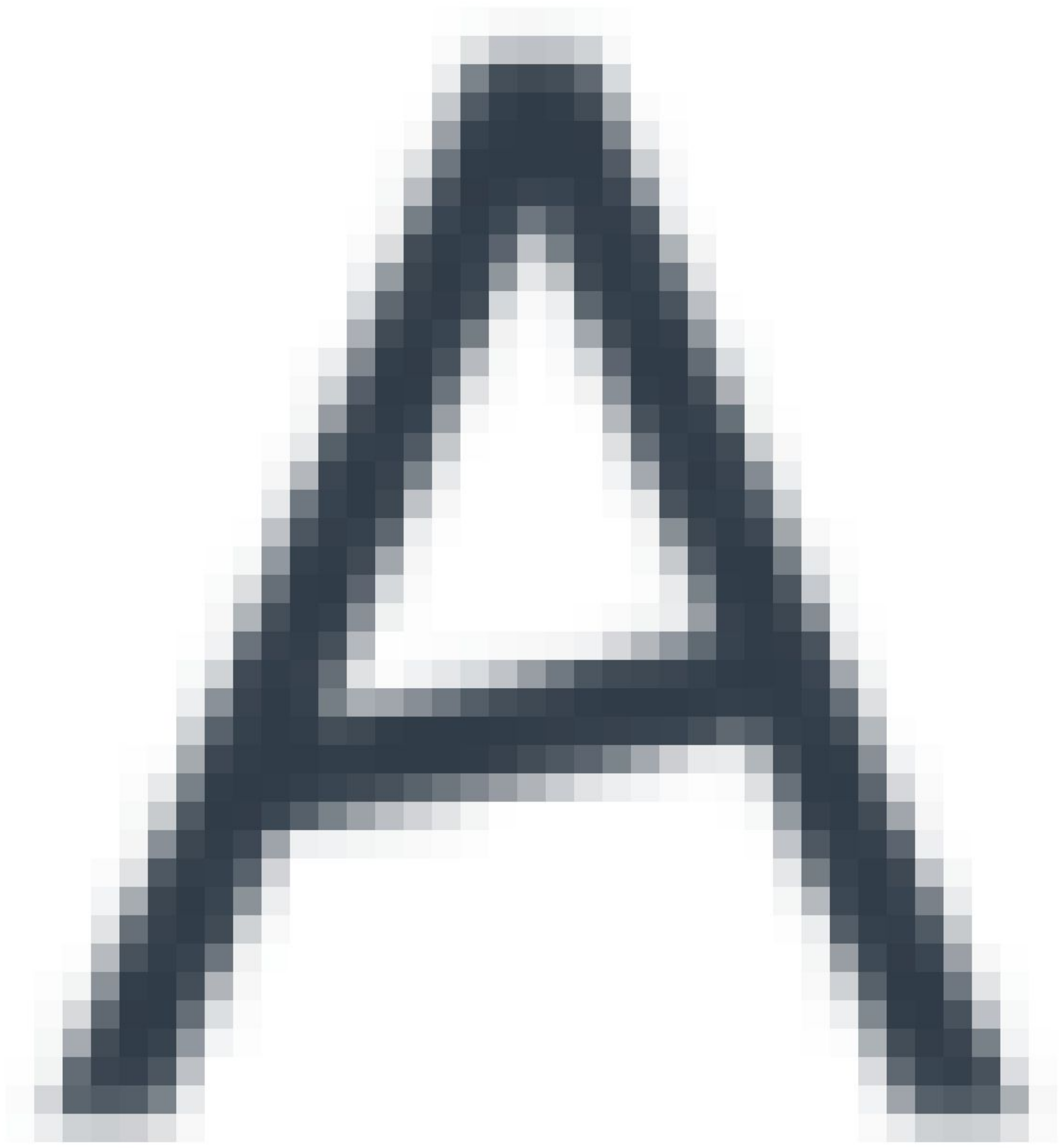
NEXT

mage Data

Images are another example of a data type that is commonly used as input in machine learning problems—but that isn't initially in numerical format. So, how do we represent an image as numbers? Let's have a look.

Taking a Closer Look at Image Data

Let's look a little closer at how an image can be encoded numerically. If you zoom in on an image far enough, you can see that it consists of small tiles, called *pixels*:



The color of each pixel is represented with a set of values:

- In **grayscale images**, each pixel can be represented by a **single** number, which typically ranges from 0 to 255. This value determines how dark the pixel appears (e.g., `0` is black, while `255` is bright white).

- In **colored images**, each pixel can be represented by a vector of **three** numbers (each ranging from 0 to 255) for the three primary color channels: red, green, and blue. These three red, green, and blue (RGB) values are used together to decide the color of that pixel. For example, purple might be represented as `128, 0, 128` (a mix of moderately intense red and blue, with no green).

The number of channels required to represent the color is known as the **color depth** or simply **depth**. With an *RGB image*, `depth = 3`, because there are three channels (Red, Green, and Blue). In contrast, a grayscale image has `depth = 1`, because there is only one channel.

Encoding an Image

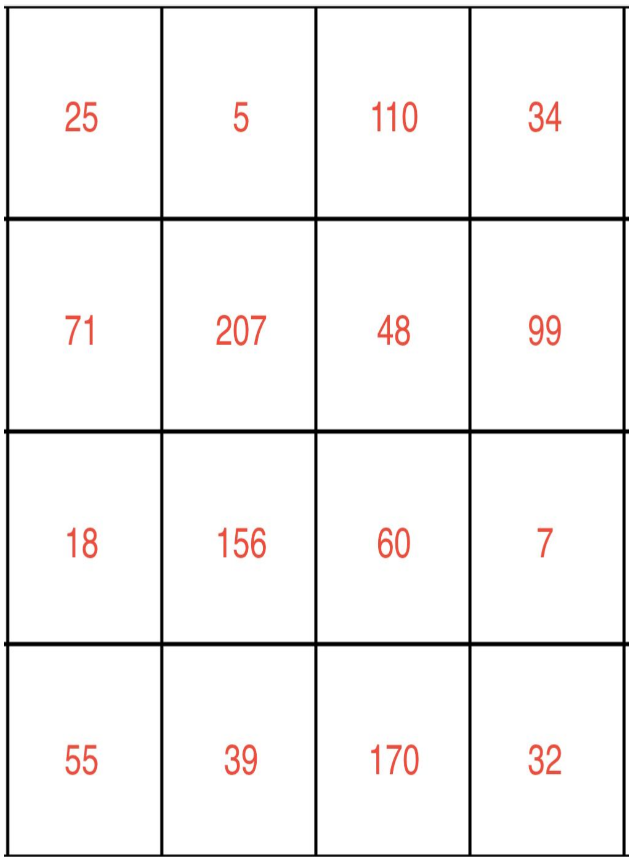
Let's now talk about how we can use this data to encode an image. We need to know the following three things about an image to reproduce it:

- Horizontal position of each pixel
- Vertical position of each pixel
- Color of each pixel

Thus, we can fully encode an image numerically by using a vector with three dimensions. The size of the vector required for any given image would be the `height * width * depth` of that image.

QUESTION 1 OF 2

Assume this figure is the *numerical representation* of an **RGB image** in the red channel:



25	5	110	34
71	207	48	99
18	156	60	7
55	39	170	32

Each of the squares represents one pixel and the value in the square is the pixel value.

Which of these statements is **incorrect**?

- This image can be encoded by a vector with the dimension of `4*4*3`
- **The total number of pixels in this image is 48**
- The numerical representation of the image in the **green channel** has the dimension of `4*4`
- The image has uniform aspect ratio but may need to be normalized.

SUBMIT

QUESTION 2 OF 2

There is a square shaped **RGB image** that consists of 900 pixels. Which of the following statements are correct?

- Without any preprocessing, the image can be encoded by a 3-dimension vector with the dimension `45*20*3`
- **This image has a dimension of `30*30`**
- If the image is cropped to half of the original size, it can be encoded by a vector with the dimension `15*15*2`

- If the image is converted to grayscale, it can be encoded by a vector with the dimension `30*30*1`

SUBMIT

Other Preprocessing Steps

In addition to encoding an image numerically, we may also need to do some other preprocessing steps. Generally, we would want to ensure that the input images have a *uniform aspect ratio* (e.g., by making sure all of the input images are square in shape) and are *normalized* (e.g. subtract mean pixel value in a channel from each pixel value in that channel). Some other preprocessing operations we might want to do to clean the input images include rotation, cropping, resizing, denoising, and centering the image.

Text Data

Text is another example of a data type that is initially non-numerical and that must be processed before it can be fed into a machine learning algorithm. Let's have a look at some of the common tasks we might do as part of this processing.

Normalization

One of the challenges that can come up in text analysis is that there are often multiple forms that mean the same thing. For example, the verb `to be` may show up as `is`, `am`, `are`, and so on. Or a document may contain alternative spellings of a word, such as `behavior` vs. `behaviour`. So one step that you will sometimes conduct in processing text is *normalization*.

*Text **normalization** is the process of transforming a piece of text into a canonical (official) form.*

Lemmatization is an example of normalization. A **lemma** is the dictionary form of a word and **lemmatization** is the process of reducing multiple inflections to that single dictionary form. For example, we can apply this to the `is`, `am`, `are` example we mentioned above:

Original word	Lemmatized word
is	be
are	be
am	be

In many cases, you may also want to remove *stop words*. **Stop words** are high-frequency words that are unnecessary (or unwanted) during the analysis. For example, when you enter a query like `which cookbook has the best pancake recipe` into a search engine, the words `which` and `the` are far less relevant than `cookbook`, `pancake`, and `recipe`. In this context, we might want to consider `which` and `the` to be *stop words* and remove them prior to analysis.

Here's another example:

Original text	Normalized text
The quick fox.	[quick, fox]
The lazzy dog.	[lazy, dog]
The rabid hare.	[rabid, hare]

Here we have **tokenized** the text (i.e., split each string of text into a list of smaller parts or *tokens*), removed stop words (`the`), and standardized spelling (changing `lazzy` to `lazy`).

QUESTION 1 OF 5

Here's another example:

Original text	Normalized text
Mary had a little lamb.	[Mary, have, a, little, lamb]
Jack and Jill went	[Jack, and, Jill, go, up, the, hill]

up the hill.	
London bridge is falling down.	[London, bridge, be, fall, down]

Looking at the normalized text, which of the following have been done?

- **Tokenization**
- Removal of *stop words*
- **Lemmatization**

SUBMIT

Vectorization

After we have normalized the text, we can take the next step of actually encoding it in a numerical form. The goal here is to identify the particular features of the text that will be relevant to us for the particular task we want to perform—and then get those features extracted in a numerical form that is accessible to the machine learning algorithm. Typically this is done by text **vectorization**—that is, by turning a piece of text into a vector. Remember, a *vector* is simply an array of numbers—so there are many different ways that we can vectorize a word or a sentence, depending on how we want to use it. Common approaches include:

- Term Frequency-Inverse Document Frequency (TF-IDF) vectorization
- Word embedding, as done with Word2vec or Global Vectors (GloVe)

The details of these approaches are a bit outside the scope of this class, but let's take a closer look at TF-IDF as an example. The approach of TF-IDF is to give less importance to words that contain less information and are common in documents, such as "the" and "this"—and to give higher importance to words that contain relevant information and appear less frequently. Thus TF-IDF assigns weights to words that signify their relevance in the documents.

Here's what the word importance might look like if we apply it to our example

quick	fox	lazy	dog	rabid	hare	the
0.32	0.23	0.12	0.23	0.56	0.12	0.0

Here's what that might look like if we apply it to the normalized text:

	quick	fox	lazy	dog	rabid	hare
[quick, fox]	**0.32**	**0.23**	0.0	0.0	0.0	0.0
[lazy, dog]	0.0	0.0	**0.12**	**0.23**	0.0	0.0

[rabid, hare]	0.0	0.0	0.0	0.0	**0.56**	**0.12**

Noticed that "the" is removed since it has `0` importance here.

Each chunk of text gets a vector (represented here as a row in the table) that is the length of the total number of words that we are interested in (in this case, six words). If the normalized text does not have the word in question, then the value in that position is `0`, whereas if it *does* have the word in question, it gets assigned to the importance of the word.

QUESTION 2 OF 5

Let's pause to make sure this idea is clear. In the table above, what does the value `0.56` mean?

- It means that the word `fox` has some importance in `[quick, fox]`.
- It means that the word `rabid` has some importance in `[quick, fox]`.
- It means that the word `fox` has some importance in `[rabid, hare]`.

- **It means that the word `rabid` has some importance in `[rabid, hare]`.**

SUBMIT

QUESTION 3 OF 5

What vector will be used to represent "quick, lazy hare"

- (0.32, 0.23, 0.12, 0.0, 0.0, 0.0)
- **(0.32, 0.0, 0.12, 0.0, 0.0, 0.12)**
- (0.0, 0.0, 0.12, 0.0, 0.56, 0.12)
- (0.0, 0.0, 0.12, 0.23, 0.0, 0.12)

SUBMIT

Feature Extraction

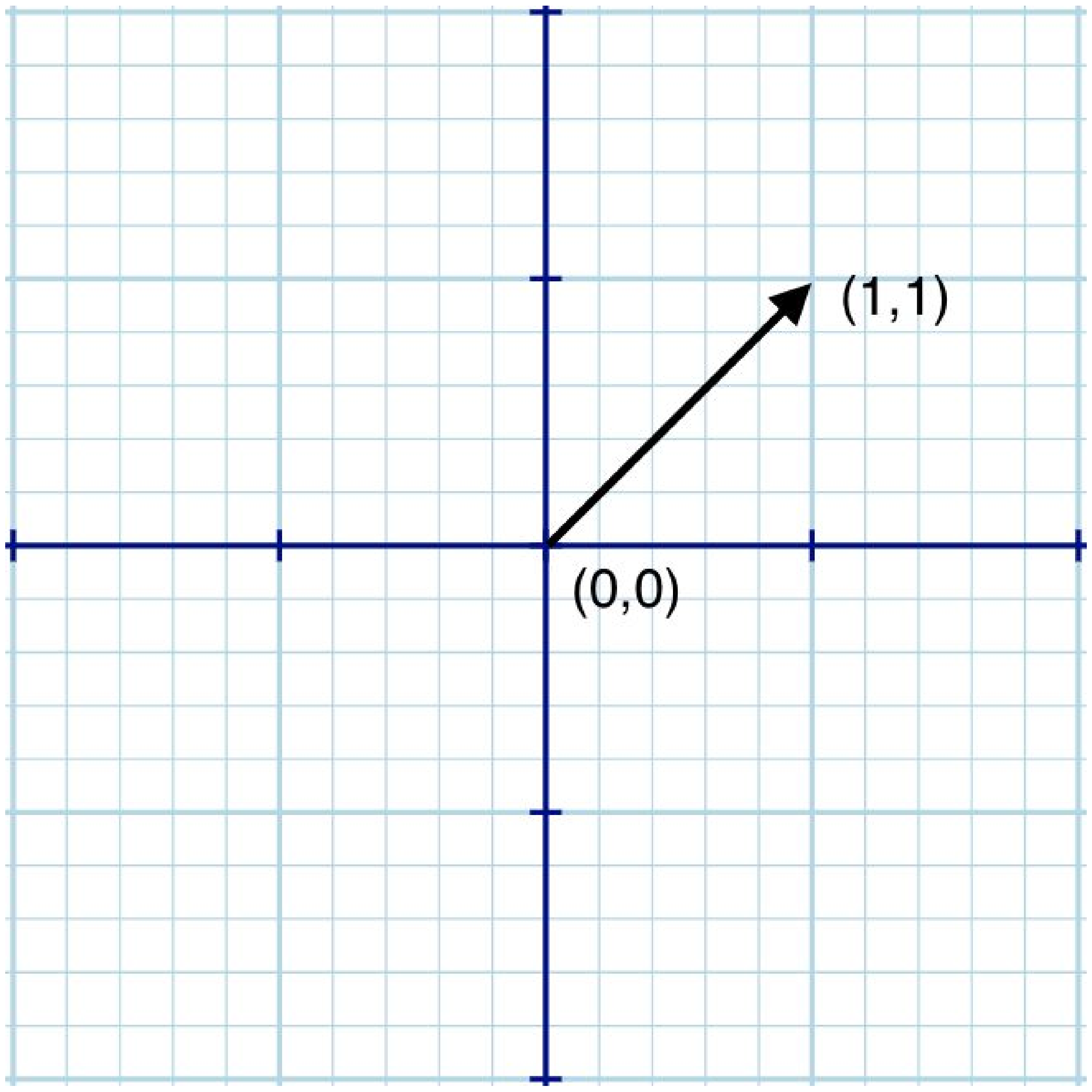
As we talked about earlier, the text in the example can be represented by vectors with length `6` since there are `6` words total.

[quick, fox] as `(0.32, 0.23, 0.0, 0.0, 0.0, 0.0)`

[lazy, dog] as `(0.0, 0.0, 0.12, 0.23, 0.0, 0.0)` [rabid, hare] as `(0.0, 0.0, 0.0 , 0.0, 0.56, 0.12)`

We understand the text because each word has a meaning. But how do algorithms understand the text using the vectors, in other words, how do algorithms extract features from the vectors?

Vectors with length `n` can be visualized as a line in an `n` dimension space. For example, a vector `(1,1)` can be viewed as a line starting from (0, 0) and ending at `(1,1)`.



Any vector with the **same length** can be visualized in the **same space**. How close one vector is to another can be calculated as **vector distance**. If two vectors are close to each other, we can say the text represented by the two vectors have a similar meaning or have some connections. For example, if we add [lazy, fox] to our example:

	quick	fox	lazy	dog	rabid	hare
[quick, fox]	**0.32**	**0.23**	0.0	0.0	0.0	0.0
[lazy, dog]	0.0	0.0	**0.12**	**0.23**	0.0	0.0
[rabid, hare]	0.0	0.0	0.0	0.0	**0.56**	**0.12**
[lazy, fox]	0.0	**0.23**	**0.12**	0.0	0.0	0.0

Apparently, [lazy, fox] is more similar to [lazy, dog] than [rabid, hare], so the vector distance of [lazy, fox] and [lazy, dog] is smaller than that to [lazy, fox] and [rabid, hare].

QUESTION 4 OF 5

Imagine the words "monkey", "rabbit", "bird" and "raven" are represented by vectors with the same length. Based on the meanings of the words, which two words would we expect to have the smallest vector distance?

- "monkey" and "rabbit"
- "monkey" and "raven"
- "rabbit" and "bird"
- **"raven" and "bird"**

SUBMIT

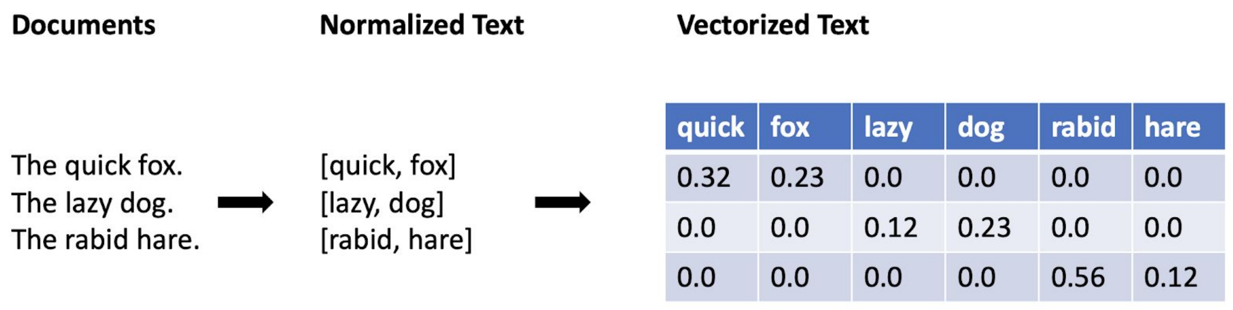
The Whole Pipeline

In this next video, we'll first review the above steps—normalization and vectorization—and then talk about how they fit into the larger goal of training a machine-learning model to analyze text data.

In summary, a typical pipeline for text data begins by pre-processing or *normalizing* the text. This step typically includes tasks such as breaking the text into sentence and word *tokens*, standardizing the spelling of words, and removing overly common words (called *stop words*).

The next step is *feature extraction and vectorization*, which creates a numeric representation of the documents. Common approaches include TF-IDF vectorization, Word2vec, and Global Vectors (GloVe).

Last, we will feed the vectorized document and labels into a model and start the training.



quick	fox	lazy	dog	rabid	hare
0.32	0.23	0.0	0.0	0.0	0.0
0.0	0.0	0.12	0.23	0.0	0.0
0.0	0.0	0.0	0.0	0.56	0.12

QUESTION 5 OF 5

What is the typical pipeline for a classification model using text data?

- vectorize text > normalize text > train model > deploy model
- train model > normalize text > vectorize text > deploy model

- vectorize text > normalize text > deploy model > train model
- **normalize text > vectorize text > train model > deploy model**

SUBMIT

NEXT

Two Perspectives on ML

Computer science vs. Statistical perspective

As you can see, data plays a central role in how problems are modeled in machine learning. In very broad terms, we can think of machine learning as a matter of using some data (perhaps historical data that we already have on hand) to train a model. Then, once the model is trained, we can feed it new input data and have it tell us something useful.

So the general idea is that we create models and then feed data into these models to generate outputs. These outputs might be, for example, predictions for future trends or patterns in the data.

This idea draws on work not only from *computer science*, but also *statistics*—and as a result, you will often see the same underlying machine learning concepts described using different terms. For example, a computer scientist might say something like:

*We are using **input features** to create a **program** that can generate the desired **output**.*

In contrast, someone with a background in statistics might be inclined to say something more like:

*We are trying to find a **mathematical function** that, given the values of the **independent variables** can predict the values of the **dependent variables**.*

While the terminology are different, the challenges are the same, that is how to get the best possible outcome.

QUIZ QUESTION

Can you match the terms below, from the computer science perspective, with their counterparts from the statistical perspective?

Submit to check your answer choices!

COMPUTER SCIENCE

STATISTICAL

program

function

input

independent variable

output

dependent variable

SUBMIT

In the end, having an understanding of the underlying concepts is more important than memorizing the terms used to describe those concepts. However, it's still essential to be familiar with the terminology so that you don't get confused when talking with people from different backgrounds.

Over the next couple of pages, we'll take a look at these two different perspectives and get familiar with some of the related terminology.

The Computer Science Perspective

Computer science terminology

As we discussed earlier, one of the simplest ways we can organize data for machine learning is in a table, like the table of clothing products we looked at earlier in this lesson:

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

What are some of the terms we can use to describe this data?

For the **rows** in the table, we might call each row an **entity** or an **observation** about an entity. In our example above, each *entity* is simply a product, and when we speak of an *observation*, we are simply referring to the data collected about a given product. You'll also sometimes see a row of data referred to as an **instance**, in the sense that a row may be considered a single example (or instance) of data.

For the **columns** in the table, we might refer to each column as a **feature** or **attribute** which describes the property of an entity. In the above example, `color` and `quantity` are *features* (or *attributes*) of the products.

Input and output

Remember that in a typical case of machine learning, you have some kind of *input* which you feed into the machine learning algorithm, and the algorithm produces some *output.* In most cases, there are multiple pieces of data being used as input. For example, we can think of a single row from the above table as a *vector* of data points:

```
(908721, Guess, Blue, 789, 45.33)
```

Again, in computer science terminology, each element of the input vector (such as `Guess` or `Blue`) is referred to as an *attribute* or *feature*. Thus, we might feed these *input features* into our machine learning program and the program would then generate some kind of desired output (such as a prediction about how well the product will sell). This can be represented as:

```
Output = Program(Input Features)
```

An important step in preparing your data for machine learning is *extracting* the relevant features from the raw data. (The topic of *feature extraction* is an important one that we'll dive into in greater detail in a later lesson.)

QUESTION 1 OF 2

Have a look at this data:

ID	Name	Species	Age
1	Jake	Cat	3
2	Bailey	Dog	7
3	**Jenna**	**Dog**	**4**
4	Marco	Cat	12

Which of the following terms might we use to refer to the part of the table that is highlighted?

(Select all that apply.)

- **A *row***
- An *attribute*
- **An *entity***
- **An *instance***
- **An *input vector***
- **A *feature***

SUBMIT

QUESTION 2 OF 2

And how about now?

ID	Name	Species	Age
1	Jake	**Cat**	3

2	Bailey	**Dog**	7
3	Jenna	**Dog**	4
4	Marco	**Cat**	12

Which of the following terms might we use to refer to the part of the table that is highlighted?

(Select all that apply.)

- **A *column***
- **An *attribute***
  - An *entity*
  - An *instance*
- **A *feature***

SUBMIT

NEXT

The Statistical Perspective

Statistical terminology

In statistics, you'll also see the data described in terms of **independent variables** and **dependent variables**. These names come from the idea that the value of one variable may *depend* on the value of some other variables. For example, the selling `price` of a house is the dependent variable that *depends* on some independent variables—like the house's `location` and `size`.

In the example of clothing products we looked at earlier in this lesson:

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

We might use data in each row (e.g. `(908721, Guess, Blue, 789, 45.33)`) to predict the sale of the corresponding item. Thus, the sale of each item is *dependent* on the data in each row. We can call the data in each row the independent variables and call the sale the dependent variable.

Input and output

From a statistical perspective, the machine learning algorithm is trying to learn a hypothetical function `(f)` such that:

```
Output Variable = f(Input Variables)
```

Typically, the *independent variables* are the input, and the *dependent variables* are the output. Thus, the above formula can also be expressed as:

```
Dependent Variable = f(Independent Variables)
```

In other words, we are feeding the independent variables into the function, and the function is giving us the resulting values of the dependent variables. With the housing example, we might want to have a function that can take the independent variables of `size` and `location` as input and use these to predict the likely selling `price` of the house as output.

Yet another way to represent this concept is to use shorthand notation. Often, the input variables are denoted as `X` and the output variable is denoted as `Y`:

```
Y = f(X)
```

In the case of multiple input variables, X would be an **input vector**, meaning that it would be composed of multiple individual inputs (e.g. `(908721, Guess, Blue, 789, 45.33)`). When this is the case, you'll see the individual inputs denoted with a subscript, as in X_1, X_2, X_3, and so on.

NEXT

The Tools for Machine Learning

Many tools have been developed to make machine learning more powerful and easier to implement. On this page, we'll take a look at the typical components you might employ in a machine learning ecosystem. You don't need to understand the details of these tools at this stage, and we don't assume you've had previous experience with them. Our goal at this point is simply to give you some idea of what some of the popular tools are and how they relate to one another.

The Machine Learning Ecosystem

A typical machine learning ecosystem is made up of three main components:

1. Libraries. When you're working on a machine learning project, you likely will not want to write all of the necessary code yourself—instead, you'll want to make use of code that has already been created and refined. That's where libraries come in. A *library* is a collection of pre-written (and compiled) code that you can make use of in your own project. *NumPy* is an example of a library popularly used in data science, while *TensorFlow* is a library specifically designed for machine learning. Read this article for some other useful library.

2. Development environments. A *development environment* is a software application (or sometimes a group of applications) that provide a whole suite of tools designed to help you (as the developer or machine learning engineer) build out your projects. *Jupyter Notebooks* and *Visual Studio* are examples of development environments that are popular for coding many different types of projects, including machine learning projects.

3. Cloud services. A *cloud service* is a service that offers data storage or computing power over the Internet. In the context of machine learning, you can use a cloud service to access a server that is likely far more powerful than your own machine, or that comes equipped with machine learning models that are ready for you to use. Read more information about different cloud services from this article

For each of these components, there are multiple options you can choose from. Let's have a look at some examples.

Notebooks

Notebooks are originally created as a documenting tool that others can use to reproduce experiments. Notebooks typically contain a combination of runnable code, output, formatted text, and visualizations. One of the most popular open-source notebooks used today by data scientists and data science engineers is **Jupyter notebook**, which can combine code, formatted text (markdown) and visualization.

Notebooks contains several independent **cells** that allow for the execution of code snippets within those cells. The output of each cell can be saved in the notebook and viewed by others.

End-to-end with Azure

You can analyze and train a small amount of data with your local machine using Jupyter notebook, Visual studio, or other tools. But with very large amounts of data, or you need a faster processor, it's a better idea to train and test the model *remotely* using *cloud services* such as Microsoft Azure. You can use Azure Data Science Virtual Machine, Azure Databricks, Azure Machine Learning Compute, or SQL server ML services to train and test models and use Azure Kubernetes to deploy models.

QUIZ QUESTION

Below are the development environments we just discussed. Can you match each one with its description?

Submit to check your answer choices!

DESCRIPTION	DEVELOPMENT ENVIRONMENT
Microsoft's core development environment	Visual Studio
Open-source tool that can combine code, markdown, and visualizations together in a single document.	Jupyter Notebooks
A light-weight code editor from Microsoft	

Visual Studio Code

Data analytics platform, optimized for use with Microsoft cloud services

Azure Databricks

SUBMIT

NEXT

Libraries for Machine Learning

For your reference, here are all the libraries we went over in the video. This is a lot of info; you should not feel like you need to be deeply knowledgable about every detail of these libraries. Rather, we suggest that you become familiar with what each library is *for*, in general terms. For example, if you hear someone talking about *matplotlib*, it would be good for you to recognize that this is a popular library for data visualization. Or if you see a reference to *TensorFlow*, it would be good to recognize this as a popular machine learning library.

Core Framework and Tools

- Python is a very popular high-level programming language that is great for data science. Its ease of use and wide support within popular machine learning platforms, coupled with a large catalog of ML libraries, has made it a leader in this space.
- Pandas is an open-source Python library designed for analyzing and manipulating data. It is particularly good for working with tabular data and time-series data.
- NumPy, like Pandas, is a Python library. NumPy provides support for large, multi-dimensional arrays of data, and has many high-level mathematical functions that can be used to perform operations on these arrays.

Machine Learning and Deep Learning

- Scikit-Learn is a Python library designed specifically for machine learning. It is designed to be integrated with other scientific and data-analysis libraries, such as NumPy, SciPy, and matplotlib (described below).
- Apache Spark is an open-source analytics engine that is designed for cluster-computing and that is often used for large-scale data processing and big data.
- TensorFlow is a free, open-source software library for machine learning built by Google Brain.

- Keras is a Python deep-learning library. It provide an Application Programming Interface (API) that can be used to interface with other libraries, such as TensorFlow, in order to program neural networks. Keras is designed for rapid development and experimentation.
- PyTorch is an open source library for machine learning, developed in large part by Facebook's AI Research lab. It is known for being comparatively easy to use, especially for developers already familiar with Python and a Pythonic code style.

Data Visualization

- Plotly is not itself a library, but rather a company that provides a number of different front-end tools for machine learning and data science—including an open source graphing library for Python.
- Matplotlib is a Python library designed for plotting 2D visualizations. It can be used to produce graphs and other figures that are high quality and usable in professional publications. You'll see that the Matplotlib library is used by a number of other libraries and tools, such as SciKit Learn (above) and Seaborn (below). You can easily import Matplotlib for use in a Python script or to create visualizations within a Jupyter Notebook.
- Seaborn is a Python library designed specifically for data visualization. It is based on matplotlib, but provides a more high-level interface and has

additional features for making visualizations more attractive and informative.

- Bokeh is an interactive data visualization library. In contrast to a library like matplotlib that generates a static image as its output, Bokeh generates visualizations in HTML and JavaScript. This allows for web-based visualizations that can have interactive features.

QUIZ QUESTION

Below are some of the libraries we just went over. See if you can match each library with its main focus.

Submit to check your answer choices!

LIBRARY

WHAT IS IT FOR?

TensorFlow

Machine learning

Matplotlib

Data visualization

Pandas	Analyzing/manipulating data
PyTorch	Machine learning
Bokeh	Data visualization

SUBMIT

NEXT

Cloud Services for Machine Learning

A typical cloud service for machine learning provides support for managing the core assets involved in machine learning projects. For your reference, you can see a table summarizing these main **assets** below. We'll explore all of these components in more detail as we go through the course.

Feature	Description
Datasets	Define, version, and monitor datasets used in machine learning runs.
Experiments / Runs	Organize machine learning workloads and keep track of each task executed through the service.

Pipelines	Structured flows of tasks to model complex machine learning flows.
Models	Model registry with support for versioning and deployment to production.

Endpoints	Expose real-time endpoints for scoring as well as pipelines for advanced automation.

Machine learning cloud services also need to provide support for **managing** the resources required for running machine learning tasks:

Feature	**Description**
Compute	Manage compute resources used by machine

	learning tasks.
Environments	Templates for standardized environments used to create compute resources.
Datastores	Data sources connected to the service environment (e.g. blob stores, file shares, Data Lake

	stores, databases).

A Brief Intro to Azure Machine Learning

Below are some of the features of Azure Machine Learning that we just discussed. We'll get some hands-on experience using these features during the labs found throughout this course. For now, our goal is just to take a brief tour of the main features.

Following are some of the features in **Azure ML workspace**, a centralized place to work with all the artifacts you create:

Feature	Description

Automated ML	Automate intensive tasks that rapidly iterate over many combinations of algorithms, hyperparameters to find the best model based on the chosen metric.
Designer	A drag-and-drop tool that lets you create ML models without a single line of code.
Datasets	A place you can create datasets.

Experimen ts	A place that helps you organize your runs.
Models	A place to save all the models created in Azure ML or trained outside of Azure ML.
Endpoints	A place stores real-time endpoints for scoring and pipeline endpoints for advanced automation.

Compute	A designated compute resource where you run the training script or host the service deployment.
Datastores	An attached storage account in which you can store datasets.

QUIZ QUESTION

Below are some of the features we just went over. Can you match each one with its description?

Submit to check your answer choices!

DESCRIPTION

FEATURE

A drag-and-drop tool that lets you create machine learning models without writing any code.

The designer

A centralized place to work with all the artifacts you create.

The Azure ML workspace

A designated resource/environment where you run your training script or host your service deployment.

Compute target

Attached storage account in which you can keep data for your machine learning workspace.

Data store

SUBMIT

NEXT

Models vs. Algorithms

In machine learning, the key distinction of a model and an algorithm is:

***Models** are the **specific representations** learned from data*

Algorithms** are the processes of **learning

We can think of the algorithm as a function—we give the algorithm data and it produces a model:

$$\text{Model} = \text{Algorithm}(\text{Data})$$

$$Model=Algorithm(Data)$$

On the next page, we'll look at this distinction in the context of a concrete example: Linear regression.

More About Machine Learning Algorithms

We can think of an algorithm as a mathematical tool that can usually be represented by an equation as well as implemented in code. For example, `y`

`= Wx + b` is an algorithm that can be used to calculate y from x if the values for `W` and `b` are known. But how do we get `W` and `b`?

This is the *learning* part of machine learning; That is, *we can learn these values from training data*. For example, suppose the following data are collected:

x	y
1	1
2	2
3	3

We can plug the data into the algorithm and calculate `W = 1` and `b = 0`. We would say that that the *algorithm was run on the data and learned the values ofr `W` and `b`*. The output of the learning process is `W = 1` and `b = 0`.

Machine Learning Models

Machine learning models are *outputs* or *specific representations* of algorithms that run on data. A model represents *what is learned* by a machine learning algorithm on the data.

In the previous example, `y = 1*x + 0` is the model we obtained from running the algorithm `y = Wx + b` on the training data. We can also say that `y = 1*x + 0` is the model that can be used to predict `y` from `x`.

A machine learning model can also be written in *a set of weights or coefficients* instead of a full equation. Looking at the previous example, since we know the algorithm, it is redundant to keep the full equation `y = 1*x + 0`. All we need are the weights (or coefficients) `W = 1` and `b = 0`. Thus, we can also think of a model as a set of weights (or coefficients) that have been learned.

QUIZ QUESTION

Which of these are examples of **algorithms** and which are examples of **models**?

Submit to check your answer choices!

EXAMPLE

ALGORITHM OR MODEL?

Least Squares Linear Regression	algorithm
Convolutional neural network	algorithm
Set of coefficients	model
An equation learned from data	model

SUBMIT

NEXT

Linear Regression

In our first lab, we're going to use Azure Machine Learning Studio to train a model using one of the fundamental machine learning algorithms: *Linear* *regression*. Before we dive into the lab, let's review what linear regression is and how it can be used to train a model.

The video below gives a brief review of the main concepts. If you've never seen linear regression before, or need a more thorough review, you can continue on for a detailed explanation below.

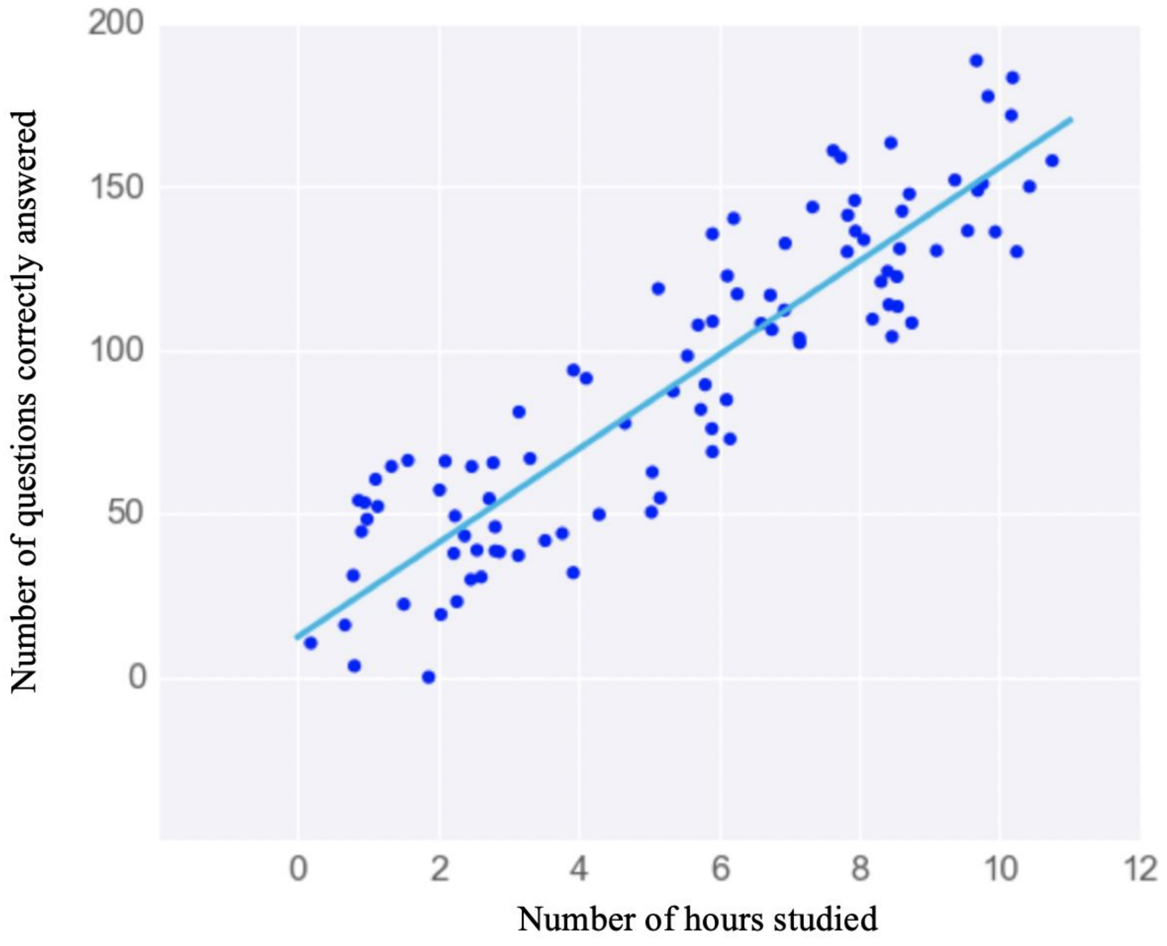
If you feel confident in your understanding of linear regression, feel free to move ahead to the next page and get started on the lab. Otherwise, we'll go over the concepts in further detail below.

Understanding Linear Regression

As the term suggests, ***linear regression*** *is an algorithm that uses a straight line (or plane) to describe relationships between variables.*

Let's consider a very simple example of a linear relationship. Suppose that we want to know if the *number of hours a student spends studying for a test* is related to the *number of questions answered correctly on the test*.

Such a relationship might look something like this:



We can see that there is a clear relationship: Students who spent more time studying also scored higher on the test.

What is more, we can see that the data points cluster around a straight line. Linear regression is all about finding the *line that best fits the data*. And this model (or line) can then be used to make predictions. In this case, if we know the number of hours a student has studied for the test, we can predict how many questions he or she can answer correctly. To make this prediction, we need the equation for the line of best fit. What would that look like?

Simple Linear Regression

You may recall from fundamental algebra that the general equation for a line looks like this:

$y = m x + b$

$y=mx+b$

Where

m

m is called the *slope* of the line, and *b* is the y-intercept. Again, this is the *general* equation. For a specific line, we need to know the values for the slope and y-intercept. For example, the following equations represent three different lines.

$y = 10 x + 50$

$y=10x+50$

$y = 2 x + 3$

$y=2x+3$

y = -10 x + 40

$y=-10x+40$

Equations like these can be used to make *predictions*. Once we know

m

m and

b

b , we can feed in a value for

x

x and the equation will give us the value of

y

y.

QUESTION 1 OF 2

For the earlier example of students studying for a test, suppose that we find that the line of best fit is:

y = 15 x + 3

$y=15x+3$

Where

x

x is the number of hours studied and

y

y is the number of questions correctly answered.

If a student studied 10 hours for the test, what is their predicted score?

- 48
- 148
- 153
- 183

SUBMIT

QUESTION 2 OF 2

Thinking back to the *models vs. algorithms* distinction, is our equation…

y = 15 x + 3

$$y=15x+3$$

…best described as a **model** or an **algorithm**?

- y = 15 x + 3
- $y=15x+3$ is a model

- y = 15 x + 3
- $y=15x+3$ is an algorithm

SUBMIT

Linear Regression in Machine Learning

The equation we used above was:

y = mx + b

$y=mx+b$

In algebraic terms, we may refer to

m

m as the **coefficient** of x or simply the **slope** of the line, and we may call

b

b the **y-intercept**. In machine learning, you will typically see the y-intercept referred to as the **bias**. In machine learning, you will also often see the equation represented using different variables, as in:

y = B_0 + B_1*x

$y=B$

0

$+B$

1

$*x$

The letters are different and the order has been changed, but it is exactly the same equation. Thus, we can see that what we know from algebra as the basic equation for a line is also, in machine learning, the equation used for **simple linear regression**.

Multiple Linear Regression

In more complex cases where there is *more than one* input variable, we might see something like this:

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

$y=B$

0

$+B$

1

$*x$

1

$+B$

2

$*x$

2

$+B$

3

$*x$

3

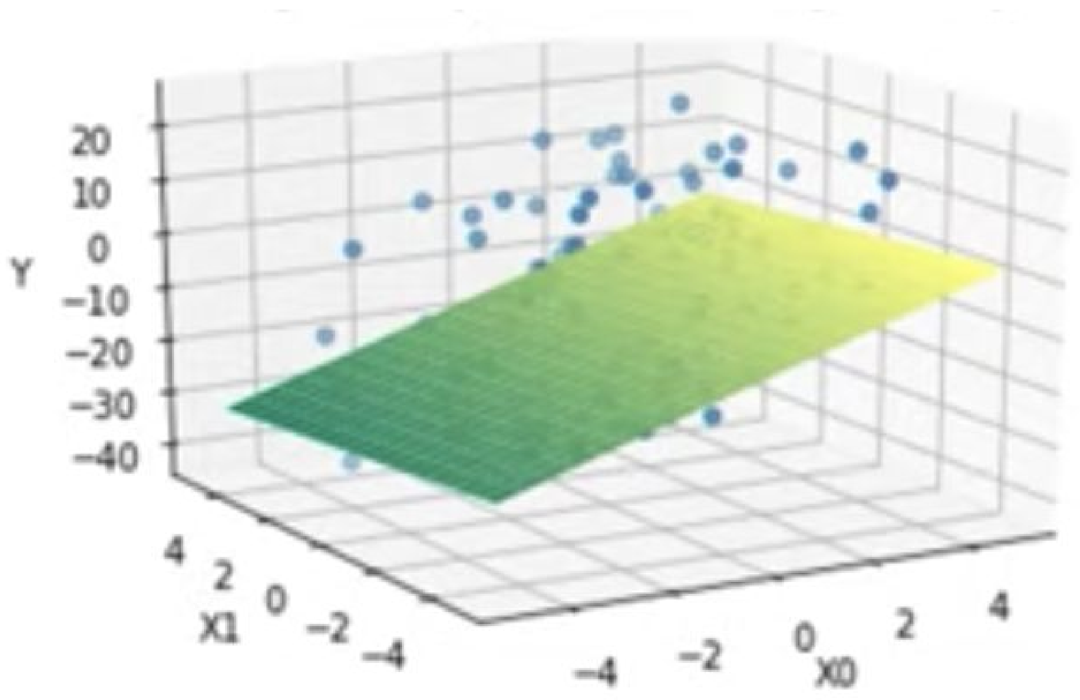
$...+B$

n

$*x$

n

In this case, we are using multiple input variables to predict the output. When we have *multiple* input variables like this, we call it **multiple linear regression**. The visualization of multiple linear regression is no longer a simple line, but instead a *plane* in multiple dimensions:



But don't let any of this intimidate you: The core idea is still that we are modeling a relationship (using a line or plane) in order to help us predict the value of some variable that we are interested in.

Training a Linear Regression Model

To "train a linear regression model" simply means to learn the *coefficients* and *bias* that *best fit the data*. This is the purpose of the linear regression algorithm. Here we will give you a high-level introduction so that you understand conceptually how it works, but we will not go into the mathematical details.

The Cost Function

Notice from our example of test scores earlier that the line we came up with did not *perfectly* fit the data. In fact, *most* of the data points were not on the line! When we predict that a student who studies for 10 hours will get a score of 153, we do not expect their score to be exactly 153. Put another way, when we make a prediction using the line, we expect the prediction to have some **error**.

The process of finding the best model is essentially a process of finding the coefficients and bias that *minimize* this error. To calculate this error, we use a **cost function.** There are many cost functions you can choose from to train a model and the resulting error will be different depending one which cost function you choose. The most commonly used cost function for linear regression is the **root mean squared error (RMSE)**

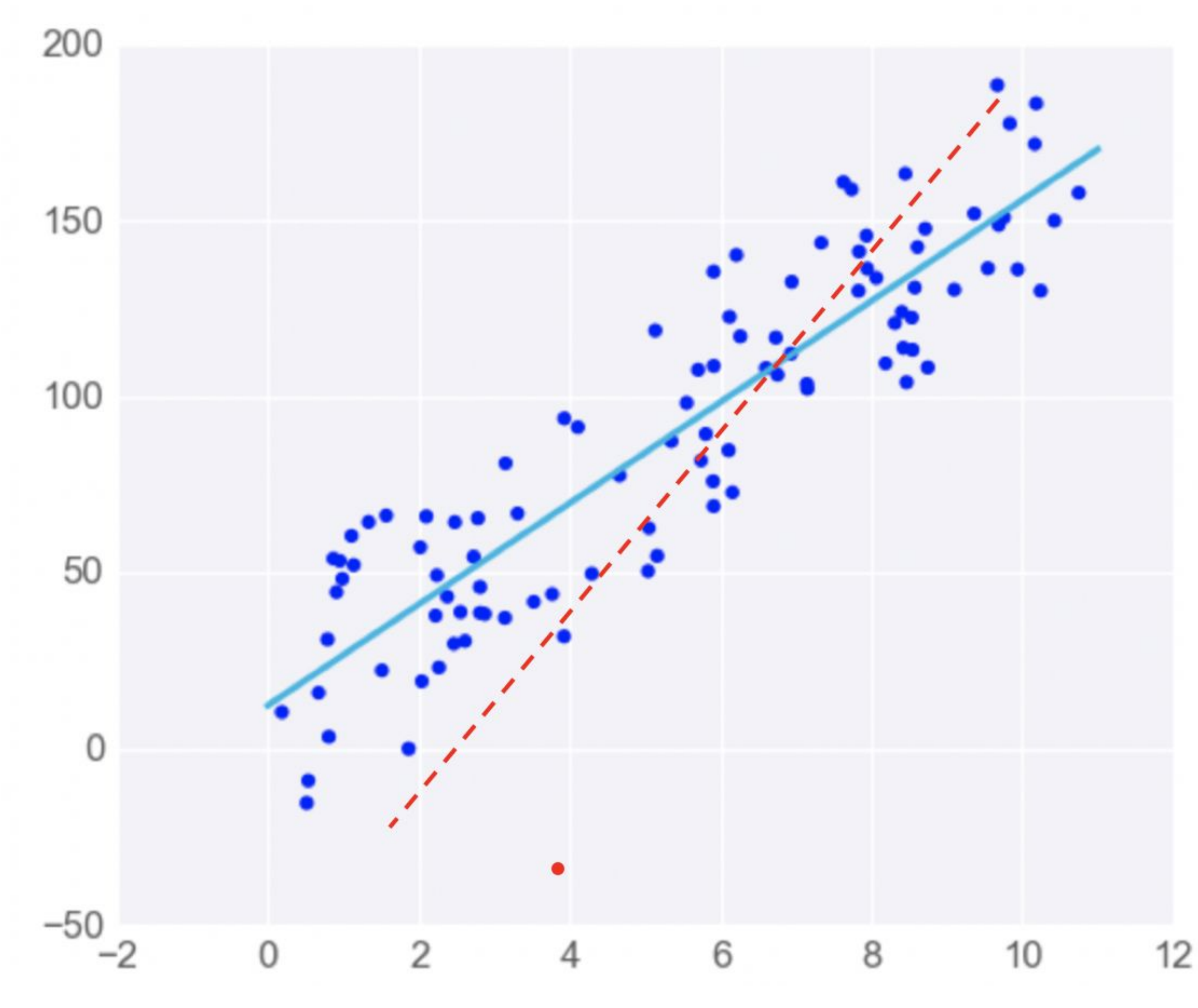
Preparing the Data

There are several **assumptions** or conditions you need to keep in mind when you use the linear regression algorithm. If the raw data does not meet these assumptions, then it needs to be prepared and transformed prior to use.

- **Linear assumption:** As we've said earlier, linear regression describes variables using a *line*. So the relationship between the input variables and the output variable needs to be a linear relationship. If the raw data does not follow a linear relationship, you may be able to transform) your data prior to using it with the linear regression algorithm. For example, if your data has an exponential relationship, you can use *log transformation*.

- **Remove collinearity**: When two variables are **collinear**, this means they can be modeled by the same line or are at least highly *correlated*; in other words, one input variable can be accurately predicted by the other. For example, suppose we want to predict education level using the input variables `number of years studying at school`, `if an individual is male`, and `if an individual is female`. In this case, we will see collinearity—the input variable `if an individual is female` can be perfectly predicted by `if an individual is male`, thus, we can say they are highly correlated. Having highly correlated input variables will make the model less consistent, so it's important to perform a *correlation check* among input variables and remove highly correlated input variables.
- **Gaussian (normal) distribution:** Linear regression assumes that the distance between output variables and real data (called *residual*) is *normally distributed*. If this is not the case in the raw data, you will need to first transform the data so that the residual has a normal distribution.
- **Rescale data:** Linear regression is very sensitive to the distance among data points, so it's always a good idea to *normalize* or *standardize* the data.

- **Remove noise**: Linear regression is very sensitive to noise and *outliers* in the data. Outliers will significantly change the line learned, as shown in the picture below. Thus, cleaning the data is a critical step prior to applying linear regression.



Calculating the Coefficients

We've discussed here the overall concept of training a linear regression model: We take the *general* equation for a line and use some data to learn the coefficients for a *specific line* that will best fit the data. Just so that you have an idea of what this looks like in concrete terms, let's look at the formulas used to calculate the coefficients. We're showing these in order to give you a general idea of what the calculations actually involve on a concrete level. For this course, you do *not* need to worry about how the formulas are derived and how to use them to calculate the coefficients.

The formula for getting the slope of the line looks something like this:

$$B1 = \frac{\sum_{i=1}^{n}(x_i - mean(x)) \times (y_i - mean(y))}{\sum_{i=1}^{n}(x_i - mean(x))^2}$$

To get the intercept, we calculate:

$$B0 = mean(y) - B1 \times mean(x)$$

And to get the *root mean squared error (RMSE)*, we have:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - y_i)^2}{n}}$$

In most machine learning libraries (such as Sklearn or Pythorch) the inner workings of the linear regression algorithm are implemented for you. The error and the best coefficients will be automatically calculated when you input the data. Here, the important thing is to understand what is happening conceptually—namely, that we choose a cost function (like RMSE) to calculate the *error* and then *minimize* that error in order to arrive at a *line of best fit* that models the training data and can be used to make predictions.

Now that we've review the concept, let's get some hands-on practice implementing the linear regression algorithm in Azure Machine Learning Studio!

NEXT

Linear Regression: Check Your Understanding

Before going on to the lab, here are some additional practice questions that you can use to check your understanding of linear regression.

QUESTION 1 OF 4

Which of the following statements about linear regression are **incorrect**?

(Select all that apply.)

- **A general equation like y = mx + b is a model**
- **Simple linear regression uses a *plane* to describe relationships between variables**
- Multiple linear regression involves more than one input variable
- Linear regression assumes that the input variables and output variable follow a linear relationship

SUBMIT

QUESTION 2 OF 4

One of the things that can be difficult when looking at a machine learning algorithm is that different terms and symbols are often used to refer to the same (or very closely related) things.

With that in mind, can you match the following terms?

Submit to check your answer choices!

LINEAR REGRESSION	MACHINE LEARNING
slope	

coefficient

intercept

bias

RMSE

cost function

SUBMIT

QUESTION 3 OF 4

Which of the following about training a linear regression model is correct?

(Select all that apply.)

- **The training process is a process of *minimizing the error***
- **A *cost function* is used to calculate the error of a model**
- A linear regression model will not change much if outliers are removed
- It is not necessary to remove highly correlated input variables when training a model
- **It is always a good idea to rescale data**

SUBMIT

QUESTION 4 OF 4

Which one of the following is not necessary when preparing data for a linear regression model?

- Remove noise
- Make sure the input variable(s) and output variable follow a linear relationship
- Remove collinearity
- **Make sure the error is minimized.**

SUBMIT

NEXT

Learning a function

As mentioned earlier, we can generally think of a machine learning *algorithm* as a process for learning, and *models* as specific representations that we train using data. In essence, machine learning algorithms aim to learn a target function (

f

f) that describes the mapping between data input variables (

X

X) and an output variable (

Y

Y).

$$\mathrm{Y = f(X)}$$

$$Y=f(X)$$

The core goal is to learn a useful transformation of the input data that gets us closer to the expected output.

Since the process extrapolates from a limited set of values, there will always be an error

e

e which is independent of the input data (

X

X) such that:

Y = f(X) + e

$Y=f(X)+e$

The variable

e

e is called irreducible error because no matter how good we get at estimating the target function (

f

f), we cannot reduce this error.

Note that the **irreducible error** we're discussing here is different from the **model error** we talked about earlier in the lesson. Irreducible error is caused by the data collection process—such as when we don't have enough data or don't have enough data features. In contrast, the model error measures how much the prediction made by the model is different from the true output. The model error is generated from the model and can be reduced during the model learning process.

QUESTION 1 OF 2

The goal of machine learning algorithms can be represented as a *learning function*:

Y = f(X) + e

$$Y=f(X)+e$$

What does each part of this represent?

Submit to check your answer choices!

PART

DESCRIPTION

Y

Y

output variable

X

X

input variables

f

f

the target function

e

e

irreducible error

SUBMIT

QUESTION 2 OF 2

Which of the following statements is most accurate about machine learning?

- Learning a function from data is usually easy, since the models produced by modern algorithms have zero error.
- It is usually clear in advance which algorithm will produce the best function.
- **In practice, it is often best to try a variety of algorithms and compare the results to see which produces the function with the least error.**

SUBMIT

NEXT

Parametric vs. Non-parametric

Based on the assumptions about the *shape* and *structure* of the function they try to learn, machine learning algorithms can be divided into two categories: **parametric** and **nonparametric**.

Parametric Machine Learning Algorithms

Parametric machine learning algorithms make assumptions about the mapping function and have a *fixed* number of parameters. No matter how much data is used to learn the model, this will not change how many parameters the algorithm has. With a parametric algorithm, we are selecting the form of the function and then learning its coefficients using the training data.

An example of this would be the approach used in linear regression algorithms, where the simplified functional form can be something like:

B_0 + B_1 * X_1 + B_2 * X_2 = 0

B

0

$+B$

1

$*X$

1

$+B$

2

$*X$

2

$=0$

This assumption greatly simplifies the learning process; after selecting the initial function, the remaining problem is simply to estimate the coefficients B0, B1, and B2 using different samples of input variables X1 and X2.

Benefits:

- **Simpler** and **easier** to understand; easier to interpret the results
- **Faster** when talking about learning from data
- **Less training data** required to learn the mapping function, working well even if the fit to data is not perfect

Limitations:

- **Highly constrained** to the specified form of the simplified function
- **Limited complexity** of the problems they are suitable for
- **Poor fit** in practice, unlikely to match the underlying mapping function.

Non-parametric Machine Learning Algorithms

Non-parametric algorithms do not make assumptions regarding the form of the mapping function between input data and output. Consequently, they are free to learn any functional form from the training data.

A simple example is the K-nearest neighbors (KNN) algorithm, which we'll discuss in more detail later in the course. KNN does not make any assumptions about the functional form, but instead uses the pattern that points have similar output when they are close.

Benefits:

- **High flexibility**, in the sense that they are capable of fitting a large number of functional forms
- **Power** by making weak or no assumptions on the underlying function
- **High performance** in the prediction models that are produced

Limitations:

- **More training data** is required to estimate the mapping function
- **Slower** to train, generally having far more parameters to train
- **Overfitting** the training data is a risk; overfitting makes it harder to explain the resulting predictions

QUESTION 1 OF 3

Which of these sentences are true statements about Machine Learning? (Select all that apply.)

- Parametric machine learning algorithms are most suitable for solving more complex problems—as opposed to nonparametric algorithms, which work great in low-complexity scenarios.

- **When there is less training data available, simplifying the form of the mapping function to a linear regression model would be the way to go.**

- **Non-parametric algorithms do not make assumptions regarding the form of the mapping between input data and output, so they are free to learn any functional form from the training data.**

SUBMIT

QUESTION 2 OF 3

For the characteristics listed below, see if you can categorize each of them as being more true about *parametric* or *non-parametric* algorithms.

Submit to check your answer choices!

CHARACTERISTIC	PARAMETRIC OR NON-PARAMETRIC?
Slower to train, generally having far more parameters	Non-parametric
Simpler and easier to understand; easier to interpret the results	Parametric
Suitable for problems of limited complexity	

Parametric

High flexibility; capable of fitting a large number of functional forms

Non-parametric

SUBMIT

QUESTION 3 OF 3

Which of the following algorithms are parametric?

(Select all that apply.)

- Decision tree
- **Logistic regression**
- KNN
- **Multiple linear regression**

SUBMIT

NEXT

Classical ML vs. Deep Learning

Remember, all deep learning algorithms are machine learning algorithms but not all machine learning algorithms are deep learning algorithms.

Deep learning algorithms are based on neural networks and the classical ML algorithms are based on classical mathematical algorithms, such as linear regression, logistic regression, decision tree, SVM, and so on.

Deep learning advantages:

- Suitable for high complexity problems
- Better accuracy, compared to classical ML
- Better support for big data
- Complex features can be learned

Deep learning disadvantages:

- Difficult to explain trained data
- Require significant computational power

Classical ML advantages:

- More suitable for small data
- Easier to interpret outcomes
- Cheaper to perform
- Can run on low-end machines
- Does not require large computational power

Classical ML disadvantages:

- Difficult to learn large datasets
- Require feature engineering
- Difficult to learn complex functions

QUESTION 1 OF 2

Which of these statements is true about classical ML vs. deep learning?

- Classical ML is a sub-category of deep learning algorithms, based on neural networks.
- **All deep learning algorithms are machine learning algorithms**
- All machine learning algorithms are deep learning algorithms

SUBMIT

QUESTION 2 OF 2

For each of the characteristics below, does it better describe *classical ML* or *deep learning*?

Submit to check your answer choices!

CHARACTERISTIC	CLASSICAL ML OR DEEP LEARNING?
Models lack transparency and are difficult to explain	Deep learning
Easier to interpret the results	Classical ML
Can learn arbitrarily complex functions from data	Deep learning
Needs explicit feature engineering	Classical ML

SUBMIT

NEXT

Approaches to Machine Learning

There are three main approaches to machine learning:

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**

We'll take a short, high-level look at these approaches here, and then revisit them in more detail in later lessons.

Supervised learning

Learns from data that contains both the inputs and expected outputs (e.g., labeled data). Common types are:

- **Classification**: Outputs are categorical.
- **Regression**: Outputs are continuous and numerical.
- **Similarity learning**: Learns from examples using a similarity function that measures how similar two objects are.
- **Feature learning**: Learns to automatically discover the representations or features from raw data.
- **Anomaly detection**: A special form of classification, which learns from data labeled as normal/abnormal.

Unsupervised learning

Learns from input data only; finds hidden structure in input data.

- **Clustering**: Assigns entities to clusters or groups.
- **Feature learning**: Features are learned from unlabeled data.
- **Anomaly detection:** Learns from unlabeled data, using the assumption that the majority of entities are normal.

Reinforcement learning

Learns how an agent should take action in an environment in order to maximize a reward function.

- **Markov decision process**: A mathematical process to model decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. Does not assume knowledge of an exact mathematical model.

The main difference between reinforcement learning and other machine learning approaches is that reinforcement learning is an *active process* where the actions of the agent influence the data observed in the future, hence influencing its own potential future states. In contrast, supervised and unsupervised learning approaches are *passive processes* where learning is performed without any actions that could influence the data.

QUESTION 1 OF 4

For each of the descriptions given below, mark which **general approach to machine learning** it best describes.

Supervised learning

DESCRIPTION

APPROACH

Learns how an agent should take actions in an environment to maximize a reward function

Reinforcement learning

Learns from data that contains only the inputs

Unsupervised learning

Learns from data that contains both the inputs and expected outputs

Supervised learning

Finds hidden structures in data

Unsupervised learning

SUBMIT

QUESTION 2 OF 4

Now let's get a bit more specific. All of the descriptions below refer to some type of supervised learning. Can you match them up?

Submit to check your answer choices!

DESCRIPTION

TYPE OF SUPERVISED LEARNING

Yields discrete categorical outputs

Classification

Learns from data labeled as normal/abnormal

Anomaly detection

Yields continuous numerical outputs

Regression

Characteristics of the data are learned using labeled data

Feature learning

Learns from examples using a similarity function

Similarity learning

SUBMIT

QUESTION 3 OF 4

All of the descriptions below refer to some type of **unsupervised learning**. Can you match them up?

Submit to check your answer choices!

DESCRIPTION

TYPE OF UNSUPERVISED LEARNING

Assigns entities to clusters or groups

Clustering

Learns from unlabeled data assuming most entities are normal

Anomaly detection

Features are learned from unlabeled data

Feature learning

SUBMIT

QUESTION 4 OF 4

Harry is an IT admin responsible for managing a legacy Web application. He has access to server performance logs with real-time system performance metrics (CPU, memory utilization, number of user sessions, number of threads, etc.). His task is to use ML to generate automated real-time alerts for preemptively detecting potential service outages.

What type of ML algorithm should Harry use?

- Supervised
- **Unsupervised**
- Reinforcement

SUBMIT

NEXT

The Trade-Offs

As all things in computer science, machine learning involves certain trade-offs. Two of the most important are **bias vs. variance** and **overfitting vs. underfitting**.

Bias vs. Variance

Bias measures how *inaccurate* the model prediction is in comparison with the true output. It is due to *erroneous assumptions* made in the machine learning process to simplify the model and make the target function easier to learn. High model complexity tends to have a low bias.

Variance measures how much the target function will *change* if different training data is used. Variance can be caused by modeling the *random noise* in the training data. High model complexity tends to have a high variance.

As a general trend, parametric and linear algorithms often have high bias and low variance, whereas non-parametric and non-linear algorithms often have low bias and high variance

Overfitting vs. Underfitting

Overfitting refers to the situation in which models fit the training data very well, but fail to generalize to new data.

Underfitting refers to the situation in which models neither fit the training data nor generalize to new data.

QUESTION 1 OF 4

Here are the main terms we just discussed. See if you can match each of them with its description.

Submit to check your answer choices!

DESCRIPTION	TERM
Error that occurs when the model is too sensitive to the training data (thus giving different estimates when given new training data)	Variance
Modeling the training data well, but not generalizing well to new data	Overfitting
failing to model the training data *and* failing to generalize to new data	Underfitting

Error that results from inaccurate assumptions in model training (that are made to simplify the training process)

Bias

SUBMIT

Bias vs. Variance Trade-off

The **prediction error** can be viewed as the sum of *model error* (error coming from the model) and the *irreducible error* (coming from data collection).

```
prediction error = Bias error + variance + error +
irreducible error
```

Low bias means *fewer* assumptions about the target function. Some examples of algorithms with low bias are KNN and decision trees. Having fewer assumptions can help generalize relevant relations between features and target outputs. In contrast, *high bias* means *more* assumptions about the target function. Linear regression would be a good example (e.g., it assumes a linear relationship). Having more assumptions can potentially miss important relations between features and outputs and cause *underfitting*.

Low variance indicates changes in training data would result in similar target functions. For example, linear regression usually has a low variance. *High variance* indicates changes in training data would result in very different target functions. For example, support vector machines usually have a high variance. High variance suggests that the algorithm learns the random noise instead of the output and causes *overfitting*.

Generally, increasing model complexity would decrease bias error since the model has more capacity to learn from the training data. But the variance error would increase if the model complexity increases, as the model may begin to learn from noise in the training data.

The goal of training machine learning models is to achieve *low bias and low variance*. The **optimal model complexity** is where bias error crosses with variance error.

Overfitting vs. Underfitting

- **k-fold cross-validation**: it split the initial training data into k subsets and train the model k times. In each training, it uses one subset as the testing data and the rest as training data.
- hold back a **validation dataset** from the initial training data to estimatete how well the model generalizes on new data.

- **simplify** the model. For example, using fewer layers or less neurons to make the neural network smaller.
- use **more data**.
- **reduce dimensionality** in training data such as PCA: it projects training data into a smaller dimension to decrease the model complexity.
- **Stop the training early** when the performance on the testing dataset has not improved after a number of training iterations.

QUESTION 2 OF 4

In machine learning, we often refer to a model as being overfitted when...

- **It is learning the training data too well at the expense of not generalizing well to new data.**
- It highly simplifies assumptions on the target function.
- It has a high variance.

SUBMIT

QUESTION 3 OF 4

Which one of the statements is true about bias and variance.

- High bias suggests fewer assumptions on the target function.

- Low variance can cause overfitting.
- **Increased model complexity generally increases variance.**
- Variance is due to erroneous assumptions on the target function.

SUBMIT

QUESTION 4 OF 4

Which technique can be used to reduce overfitting?

(Select all that apply.)

- **Have a validation dataset**
- Have less data
- **Use k-fold cross-validation**
- **Use PCA**
- Stop the training when performance on training data is stable

SUBMIT

NEXT

Lesson Summary

In this lesson, our goal was to give you a high-level introduction to the field of machine learning, including the broader context in which this branch of computer science exists.

Here are the main topics we covered:

- What machine learning is and why it's so important in today's world
- The historical context of machine learning
- The data science process
- The types of data that machine learning deals with
- The two main perspectives in ML: the *statistical* perspective and the *computer science* perspective
- The essential tools needed for designing and training machine learning models
- The basics of Azure ML
- The distinction between models and algorithms
- The basics of a linear regression model
- The distinction between parametric vs. non-parametric functions
- The distinction between classical machine learning vs. deep learning
- The main approaches to machine learning
- The trade-offs that come up when making decisions about how to design and training machine learning models

In the process, you also trained your first machine learning model using Azure Machine Learning Studio.

NEXT

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.

NEXT

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.

QUIZ QUESTION

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

SUBMIT

The best way to understanding data wrangling is to get hands-on experience actually doing it—so that's what we'll do in our next lab.

NEXT

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

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.

QUESTION 1 OF 2

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

SUBMIT

The Data Access Workflow

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 the input dataset so that you can retrain the model and maintain its accuracy.

QUESTION 2 OF 2

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

SUBMIT

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.

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.

QUIZ QUESTION

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.**

SUBMIT

NEXT

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.

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.

QUIZ QUESTION

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

SUBMIT

NEXT

Feature Engineering

QUESTION 1 OF 3

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 impleme**nted 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.**

SUBMIT

Examples of Feature Engineering Tasks

QUESTION 2 OF 3

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

SUBMIT

Summary of Feature Engineering Approaches by Data Type

QUESTION 3 OF 3

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

SUBMIT

NEXT

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.

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

QUIZ QUESTION

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)

SUBMIT

NEXT

Data Drift

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.

QUESTION 1 OF 2

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.	Upstream process changes
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).	Data quality issues

SUBMIT

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

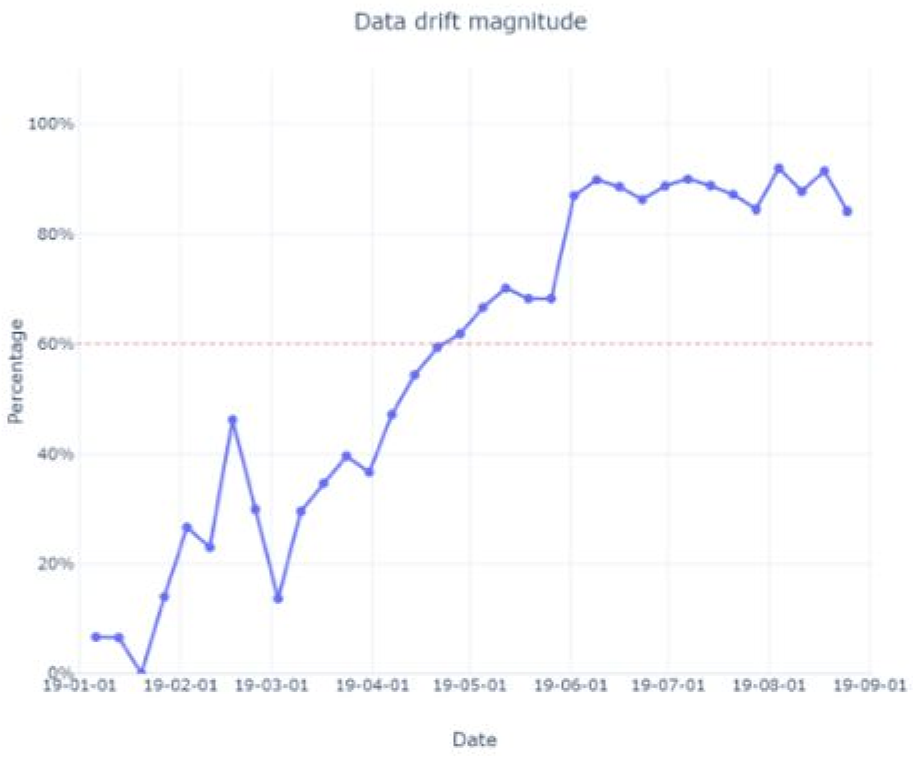
Here are a couple different types of comparisons you might want to make when monitoring for data drift:

- **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.

QUESTION 2 OF 2

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

SUBMIT

NEXT

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

X, and have it predict the value of some output feature,

y

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

$y=B$

0

$+B$

1

$*x$

1

$+B$

2

$*x$

2

$+B$

3

$*x$

3

$...+B$

n

$*x$

n

The coefficients in this equation,

B_0 … B_n

B

0

$\dots B$

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.

QUIZ QUESTION

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

SUBMIT

NEXT

Model Training in Azure Machine Learning

Taxonomy of Azure Machine Learning

QUIZ QUESTION

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.)

Model telemetry

Run

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

SUBMIT

NEXT

Training Classifiers

As we described in the last lesson, two of the main types of *supervised learning* are **classification** and **regression**. In this section, we'll get some practice training both of these types of models. But first, let's discuss the concepts in more detail—starting with **classification**.

*In a **classification** problem, the outputs are categorical or discrete.*

For example, you might want to classify emails as *spam* or *not spam*; each of these is a discrete category.

QUIZ QUESTION

As we described in the video, there are three main types of classification problem. Can you mark which type each of these examples belongs to?

Submit to check your answer choices!

EXAMPLE

TYPE

Classify an image as one (and only one) of five possible fruits.

Multi-class single-label classification

Classify medical test results as "positive" or "negative".

Binary classification

Classify music as belonging to multiple groups (e.g., "upbeat", "jazzy", "pop").

Multi-class multi-label classification

SUBMIT

NEXT

Training Regressors

Now let us turn to *regression*. The main distinction that sets a regression problem apart from a classification problem is the form of the output:

*In a **regression** problem, the output is numerical or continuous.*

A classic example would be a problem in which you are given data concerning houses and then asked to predict the price; this is a regression problem because *price* is a continuous, numerical output.

QUIZ QUESTION

Suppose that you want to predict the *probability* that a user will like classical music. What type of problem would this be?

- Classification
- Regression to arbitrary values
- **Regression to values between 0 and 1**

SUBMIT

NEXT

Evaluating Model Performance

It is not enough to simply train a model on some data and then assume that the model will subsequently perform well on future data. Instead, as we've mentioned previously, we need to split off a portion of our labeled data and reserve it for evaluating our model's final performance. We refer to this as the *test dataset*.

*The **test dataset** is a portion of labeled data that is split off and reserved for model evaluation.*

If a model learns to perform well with the training data, but performs poorly with the test data, then there may be a problem that we will need to address before putting our model out into the real world. In practice, we will also need to decide what metrics we will use to evaluate performance, and whether there are any particular thresholds that the model needs to meet on these metrics in order for us to decide that it is "good enough."

When splitting the available data, it is important to preserve the *statistical properties* of that data. This means that the data in the training, validation, and test datasets need to have similar statistical properties as the original data to prevent bias in the trained model.

QUIZ QUESTION

A researcher has collected datasets from two neighboring cities in order to develop a model that predicts the sale price of a house based on various features. Which of the following approaches would be best for this researcher?

- Split the data by city, so that data for the first city is used to train the model, and data for the second city is used to evaluate the model.

- Use all of the data for training the model, as this will result in greater power and the best possible model optimization.
- **Split the data randomly, so that some houses from each city are included in both the training dataset and the test dataset.**

SUBMIT

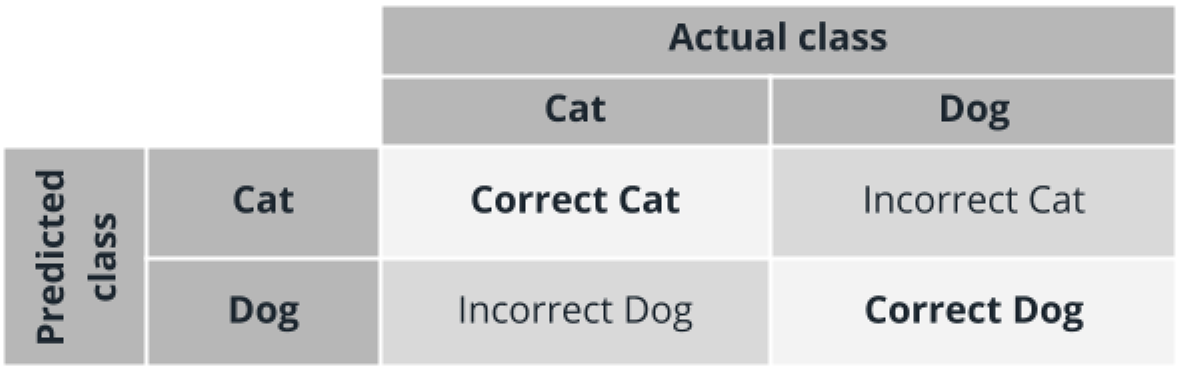
NEXT

Confusion Matrices

Suppose that we have trained a simple binary classification model: Given an image, this model will indicate whether it is a picture of a cat or a picture of a dog. How can we evaluate our model's performance? What is a good metric for doing so?

Let us first consider what it means for the model to perform well. If the model tells us an image has a dog in it *and that image actually has a dog*, we would say it performs well. And similarly, if it says that the image has a cat *and it actually has a cat* that would also be good.

To help us think about the problem, we can construct a table that shows all of the possibilities:

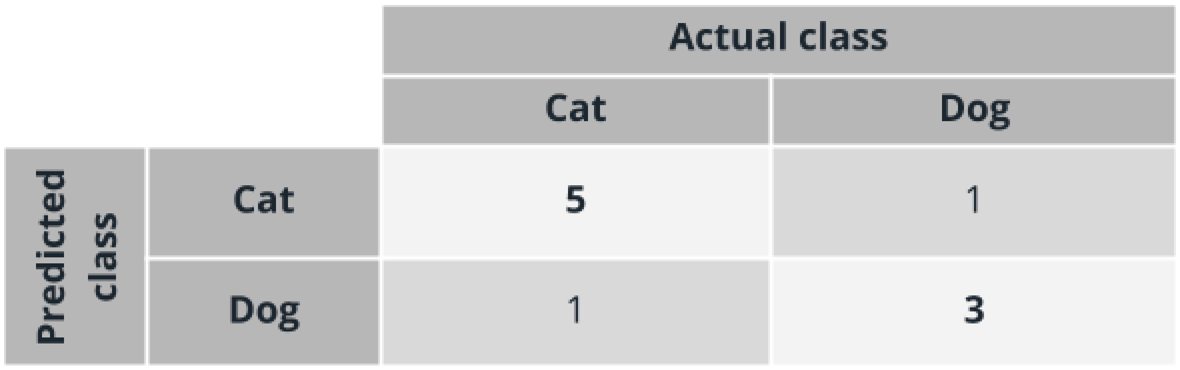


		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	**Correct Cat**	Incorrect Cat
	Dog	Incorrect Dog	**Correct Dog**

As you can see, the columns here represent the *actual class*—that is, whether an image *actually* has a dog or a cat. The rows represent the *predicted class*—that is, whether the model concludes that an image has a dog or a cat. When the predicted class matches the actual class (e.g., the model says the image has a cat and the image does indeed have a cat), this is a *correct* classification.

QUESTION 1 OF 3

We can use a table like this to organize our model's output. Suppose that we give the classifier test data containing 10 images, and the results are as follows:



		Actual class	
		Cat	Dog
Predicted class	Cat	**5**	1
	Dog	1	**3**

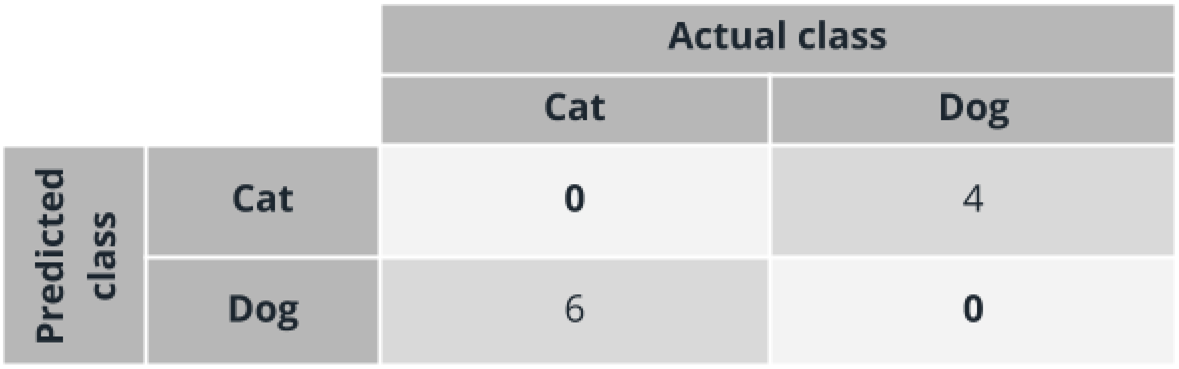
How well has the model performed?

- The model performed perfectly—there were no misclassified images at all
- **The model performed well, though not perfectly—there were a few misclassified images**
- The model performed poorly—most of the images were misclassified
- The model performed very badly—it misclassified *all* of the images

SUBMIT

QUESTION 2 OF 3

OK, here is a different set of results:



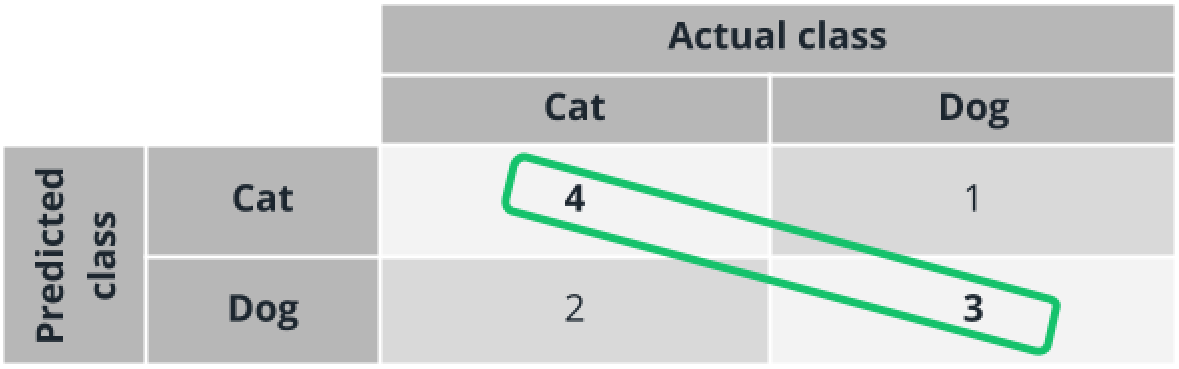
		Actual class	
		Cat	Dog
Predicted class	Cat	**0**	4
	Dog	6	**0**

How well has the model performed this time?

- The model performed perfectly—there were no misclassified images at all
- The model performed quite well, though not perfectly—there were a few misclassified images
- The model performed poorly—most of the images were misclassified
- **The model performed very badly—it misclassified *all* of the images**

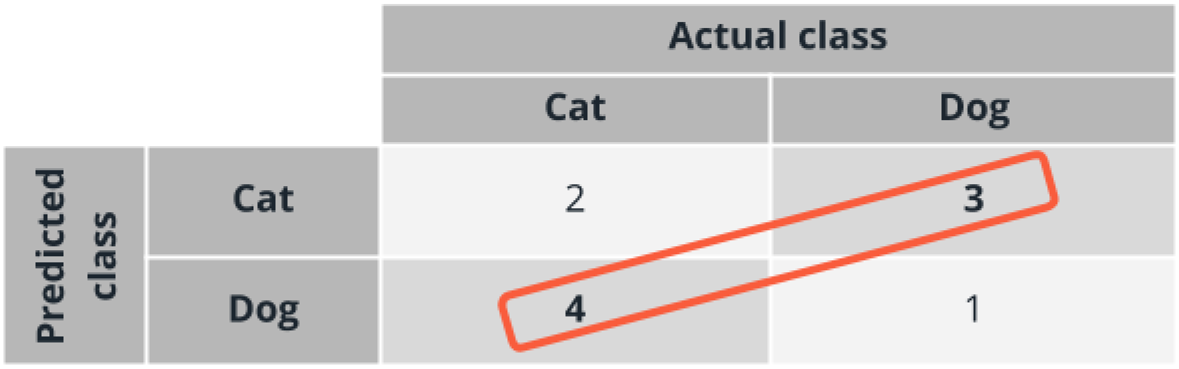
SUBMIT

The key is to look at the diagonals. If the upper left and lower right cells are high relative to the others, then the model is making more correct classifications than incorrect classifications:



		Actual class	
		Cat	Dog
Predicted class	Cat	4	1
	Dog	2	3

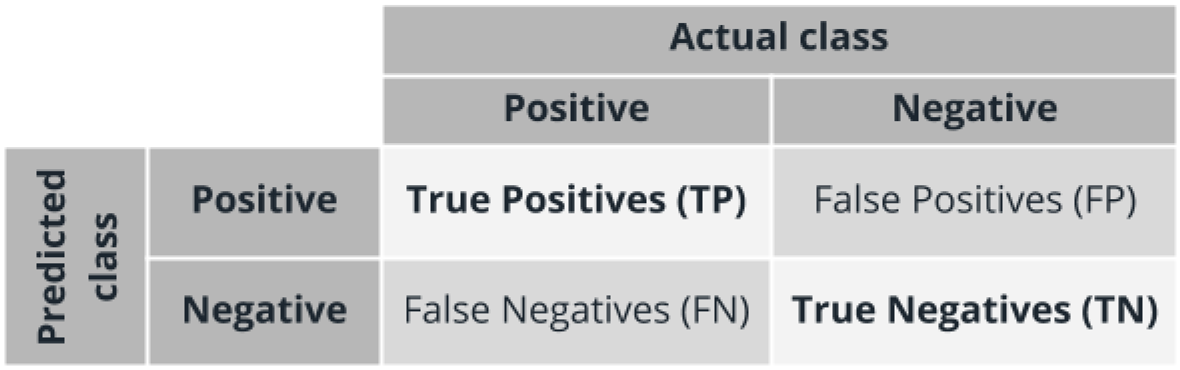
Whereas if the upper right and lower left cells are comparatively higher, the model is making more incorrect classifications:



		Actual class	
		Cat	Dog
Predicted class	Cat	2	3
	Dog	4	1

This type of table is called a **confusion matrix**. A confusion matrix gets its name from the fact that it is easy to see whether the model is getting *confused* and misclassifying the data.

You will often see the confusion matrix represented in a more general, abstract form that uses the terms *positive* and *negative*:



		Actual class	
		Positive	Negative
Predicted class	Positive	**True Positives (TP)**	False Positives (FP)
	Negative	False Negatives (FN)	**True Negatives (TN)**

- **True positives** are the *positive* cases that are *correctly* predicted as *positive* by the model
- **False positives** are the *negative* cases that are *incorrectly* predicted as *positive* by the model
- **True negatives** are the *negative* cases that are *correctly* predicted as *negative* by the model
- **False negatives** are the *positive* cases that are *incorrectly* predicted as *negative* by the model

QUESTION 3 OF 3

Which one of the following is **incorrect** about confusion matrices?

- The sum of TN and FN tells us the number of cases predicted as negative by the model
- **The sum FP and TP tells us the number of actual positive cases in the dataset**

- The sum of FP and TN tells us the number of actual negative cases in the datasets
- The sum of TP and TN tells us the number of cases correctly predicted by the model

SUBMIT

We can construct several different very useful metrics from a confusion matrix—and that's what we'll look at next.

NEXT

Evaluation Metrics for Classification

QUESTION 1 OF 2

As we just saw, the confusion matrix gives us several different metrics we can use to measure the performance of our model. See if you can remember which formula is used to calculate each metric.

Submit to check your answer choices!

FORMULA

METRIC

\frac{TP + TN}{TP+FP+FN+TN}

TP+FP+FN+TN

TP+TN

Accuracy

\frac{TP}{TP+FP}

TP+FP

TP

Precision

\frac{TP}{TP+FN}

TP+FN

TP

Recall

2*\frac{Precision * Recall}{Precision + Recall}

2*

Precision+Recall

*Precision*Recall*

F1 score

SUBMIT

Model Evaluation Charts

QUESTION 2 OF 2

In the **Receiver Operating Characteristics (ROC)** chart that we just looked at, the **Area Under the Curve (AUC)** for the diagonal line is 0.5. What does this indicate?

- A classifier that performs perfectly
- A classifier that performs moderately well
- **A classifier that performs no better than random guessing**
- A classifier that is always (or almost always) wrong

SUBMIT

NEXT

Evaluation Metrics for Regression

If you recall, classification yields discrete outputs (e.g., `cat` vs `dog` or `positive` vs. `negative`), while regression yields continuous, numerical outputs (e.g., `3.229`, `23 minutes`, `$17.78`).

Not surprisingly then, we need a different set of metrics for evaluating regression models. Let's have a look.

Again, note that with regression metrics, we are using functions that in some way calculate the numerical difference between the predicted vs. expected values.

QUIZ QUESTION

Below are the regression metrics we just discussed. Can you match each one with the description of what it measures?

Submit to check your answer choices!

WHAT IT MEASURES

METRIC

How close the regression line is to the true values.

R-Squared

Square root of the squared differences between the predicted and actual values.

RMSE

Average of the absolute difference between each prediction and the true value.

MAE

Strength and direction of the relationship between predicted and actual values.

Spearman correlation

SUBMIT

NEXT

Strength in Numbers

Remember, no matter how well-trained an individual model is, there is still a significant chance that it could perform poorly or produce incorrect results.

Rather than relying on a single model, you can often get better results by training multiple models or using multiple algorithms and in some way capturing the collective results. As we mentioned, there are two main approaches to this: **Ensemble learning** and **automated machine learning**. Let's have a closer look at each of them.

Ensemble Learning

Remember, **ensemble learning** combines multiple machine learning models to produce one predictive model. There are three main types of ensemble algorithms:

Bagging or **bootstrap aggregation**

- Helps reduce overfitting for models that tend to have high variance (such as *decision trees*)
- Uses random subsampling of the training data to produce a *bag* of trained models.
- The resulting trained models are homogeneous
- The final prediction is an average prediction from individual models

Boosting

- Helps reduce bias for models.
- In contrast to bagging, boosting uses the same input data to train multiple models using different hyperparameters.
- Boosting trains model in *sequence* by training weak learners one by one, with each new learner correcting errors from previous learners
- The final predictions are a *weighted* average from the individual models

Stacking

- Trains a large number of completely different (heterogeneous) models
- Combines the outputs of the individual models into a meta-model that yields more accurate predictions

Strength in Variety: Automated ML

Automated machine learning, like the name suggests, automates many of the iterative, time-consuming, tasks involved in model development (such as selecting the best features, scaling features optimally, choosing the best algorithms, and tuning hyperparameters). Automated ML allows data scientists, analysts, and developers to build models with greater scale, efficiency, and productivity—all while sustaining model quality.

QUIZ QUESTION

Luis has been experimenting with a machine learning algorithm that make predictions by calculating the weighted averages of weak classifiers. What is the type of machine learning algorithm Luis is working with?

- Bagging
- **Boosting**
- Stacking

SUBMIT

Now that we've talked about the concepts underlying ensemble learning and automated machine learning, let's get some hands-on practice with both of these approaches in Azure Machine Leaning Studio.

Lesson Summary

In this lesson, you've learned to perform the essential **data preparation and management** tasks involved in machine learning:

- Data importing and transformation
- The use of *datastores* and *datasets*
- Versioning

- Feature engineering
- Monitoring for data drift

The second major area we covered in this lesson was **model training**, including:

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

The final part of the lesson focused on how to get better results by using multiple trained models instead of a single one. In this context, you learned about **ensemble learning** and **automated machine learning**. You've learned how the two differ, yet apply the same general principle of "strength in numbers". In the process, you trained an ensemble model (a decision forest) and a straightforward classifier using automated Machine Learning.

NEXT

Lesson Overview

This lesson covers two of Machine Learning's fundamental approaches: **supervised** and **unsupervised** learning.

First, we'll cover **supervised learning.** Specifically, we'll learn:

- More about *classification* and *regression*, two of the most representative supervised learning tasks
- Some of the major *algorithms* involved in supervised learning, as well as how to evaluate and compare their performance
- How to use *automated machine learning* to automate the training and selection of classifiers and regressors, and how to use the Designer in Azure Machine Learning Studio to create automated Machine Learning experiments

Next, the lesson will focus on **unsupervised learning**, including:

- Its most representative learning task, *clustering*
- How unsupervised learning can address challenges like lack of labeled data, the curse of dimensionality, overfitting, feature engineering, and outliers
- An introduction to *representation learning*
- How to train your first clustering model in Azure Machine Learning Studio

NEXT

Supervised Learning: Classification

The first type of *supervised learning* that we'll look at is *classification*. Recall that the main distinguishing characteristic of classification is the type of output it produces:

In a ***classification*** *problem, the outputs are categorical or discrete.*

Within this broad definition, there are several main approaches, which differ based on how many classes or categories are used, and whether each output can belong to only one class or multiple classes. Let's have a look.

Some of the most common types of classification problems include:

- *Classification on tabular data:* The data is available in the form of rows and columns, potentially originating from a wide variety of data sources.
- *Classification on image or sound data:* The training data consists of images or sounds whose categories are already known.
- *Classification on text data:* The training data consists of texts whose categories are already known.

As we discussed in a previous lesson, machine learning requires numerical data. This means that with images, sound, and text, several steps need to be performed during the preparation phase to transform the data into numerical vectors that can be accepted by the classification algorithms.

Categories of Algorithms

QUESTION 1 OF 2

As we just discussed, at a high level there are three main categories of classification algorithms. Can you match each of them with the correct description?

Submit to check your answer choices!

DESCRIPTION

TYPE OF CLASSIFICATION

The classifier chooses from multiple categories; each output can belong to one or more categories.

Multi-class multi-label classification

The classifier chooses from multiple categories; each output belongs to single category only.

Multi-class single-label classification

The classifier choose from only two categories; each output belongs to one or the other.

Binary classification

SUBMIT

QUESTION 2 OF 2

Given the following confusion matrix:

Class	Positive	Negative
Positive	50	5
Negative	20	100

What is the value for the Precision metric?

- 0.71
- 0.95
- **0.91**
- 0.86

SUBMIT

NEXT

Multi-Class Algorithms

QUIZ QUESTION

Which of the following charts or metrics are used when evaluating results of a classification algorithm?

(Select all that apply.)

- **ROC curve**
- Mean Absolute Error
- **Confusion matrix**

- **Recall**

- Predicted vs True chart

SUBMIT

NEXT

Supervised Learning: Regression

The first type of *supervised learning* that we'll look at is *classification*. Again, the main distinguishing characteristic of regression is the type of output it produces:

*In a **regression** problem, the output is numerical or continuous.*

Introduction to Regression

Common types of regression problems include:

- *Regression on tabular data:* The data is available in the form of rows and columns, potentially originating from a wide variety of data sources.

- *Regression on image or sound data:* Training data consists of images/sounds whose numerical scores are already known. Several steps need to be performed during the preparation phase to transform images/sounds into numerical vectors accepted by the algorithms.
- *Regression on text data:* Training data consists of texts whose numerical scores are already known. Several steps need to be performed during the preparation phase to transform text into numerical vectors accepted by the algorithms.

Categories of Algorithms

Common machine learning algorithms for regression problems include:

- Linear Regression
    - Fast training, linear model
- Decision Forest Regression
    - Accurate, fast training times
- Neural Net Regression
    - Accurate, long training times

QUESTION 1 OF 2

Can you match each of these types of regression with the appropriate characteristics?

Submit to check your answer choices!

CHARACTERISTICS

REGRESSION TYPE

Accurate, fast training times

Decision Forest Regression

Accurate, long training times

Neural Net Regression

Fast training, linear model

Linear Regression

SUBMIT

QUESTION 2 OF 2

Match the following hyperparameters with their respective descriptions.

Submit to check your answer choices!

DESCRIPTION

HYPERPARAMETER

A value that defines the step taken at each iteration, before correction.	Learning rate
Penalize models to prevent overfitting.	L2 regularization weight
The maximum number of times the algorithm processes the training cases.	Number of learning iterations
A method that minimizes the amount of error at each step of the model training process.	Gradient descent

SUBMIT

NEXT

Automate the Training of Regressors

QUIZ QUESTION

Automated Machine Learning gives users the option to automatically scale and normalize input features. It also gives users the ability to enable additional featurization, such as missing values imputation, encoding, and transforms.

What is the default behavior in Automated Machine Learning to deal with missing values for categorical features?

- Impute with average value
- **Impute with most frequent value**
- Custom substitution value
- Remove entire row

SUBMIT

NEXT

Unsupervised Learning

All of the algorithms we have looked at so far are examples of *supervised learning*, in which the training data is *labeled*. For example, if we are training a classifier to recognize an image of a cat, we might have some images in our training dataset that we already know have cats—and are labeled as such.

But the cost of obtaining labeled data can be high. So now let's turn our attention to the second main type of machine learning explored in this lesson: **unsupervised learning**.

*In **unsupervised learning**, algorithms learn from unlabeled data by looking for hidden structures in the data.*

Obtaining unlabeled data is comparatively inexpensive and unsupervised learning can be used to uncover very useful information in such data. For example, we can use **clustering** algorithms to discover implicit grouping within the data, or use **association** algorithms to discover hidden rules that are governing the data (e.g., people who buy product A also tend to buy product B).

Types of Unsupervised Machine Learning

QUIZ QUESTION

Which of the following are examples of unsupervised learning algorithms?

(Select all that apply.)

- **K-Means Clustering**
- **Principal Component Analysis (PCA)**
- Support Vector Machine (SVM)
- Linear Regression
- **Autoencoders**

SUBMIT

NEXT

Semi-Supervised Learning

Sometimes fully labeled data cannot be obtained, or is too expensive—but at the same time, it may be possible to get partially labeled data. This is where **semi-supervised learning** is useful.

***Semi-supervised learning** combines the supervised and unsupervised approaches; typically it involves having small amounts of labeled data and large amounts of unlabeled data.*

QUIZ QUESTION

Which of these best describes **self training**?

- **The model is trained with the labeled data, then used to make predictions for the unlabeled data (resulting in a dataset that is fully labeled).**
- Multiple models are trained on different views of the data (e.g., different feature selections, model architectures, etc.).
- A single model is trained on different views of the data (e.g., different feature selections, model architectures, etc.).

SUBMIT

NEXT

Clustering

On this page, we'll discuss the unsupervised approach of *clustering* in more detail.

*As the name suggests, **clustering** is the problem of organizing entities from the input data into a finite number of subsets or clusters; the goal is to maximize both intra-cluster similarity and inter-cluster differences.*

Clustering Algorithms

K-Means Clustering

QUESTION 1 OF 2

Here are the main types of clustering algorithms we just discussed. Can you match each one with its description?

Submit to check your answer choices!

DESCRIPTION

TYPE OF CLUSTERING

Groups members based on how closely they are packed together; can learn clusters of arbitrary shape.

Builds a tree of clusters.

Groups members based on their distance from the center of the cluster.

Groups members based on the probability of a member belonging to a particular distribution.

SUBMIT

QUESTION 2 OF 2

Which of the following statements are true about the K-means clustering algorithm?

(Select all that apply.)

- **K-Means is a centroid-based, unsupervised clustering algorithm.**
- K-Means is a density-based, unsupervised clustering algorithm.
- **It creates up to a target (K) number of clusters and groups similar members together in a cluster.**
- The objective is to maximize intra-cluster distances

SUBMIT

NEXT

Lesson Summary

This lesson covered two of Machine Learning's fundamental approaches: **supervised** and **unsupervised** learning.

First, we learned about **supervised learning**. Specifically, we learned:

- More about *classification* and *regression*, two of the most representative supervised learning tasks
- Some of the major *algorithms* involved in supervised learning, as well as how to evaluate and compare their performance
- How to use the Designer in Azure Machine Learning Studio to build pipelines that train and compare the performance of both binary and multi-class classifiers.
- How to use *automated machine learning* to automate the training and selection of classifiers and regressors, and how to use the Designer in Azure Machine Learning Studio to create automated Machine Learning experiments

Next, the lesson focused on **unsupervised learning**, including:

- Its most representative learning task, *clustering*
- How unsupervised learning can address challenges like lack of labeled data, the curse of dimensionality, overfitting, feature engineering, and outliers
- An introduction to *representation learning*
- How to train your first clustering model in Azure Machine Learning Studio

NEXT

Lesson Overview

In this lesson, we will first look at **deep learning.** You'll learn about:

- The differences between classical machine learning and deep learning
- The benefits and applications of Deep Learning
- How to train your first neural network model

Next, you will learn about some of the most important **specialized cases of model training**, including:

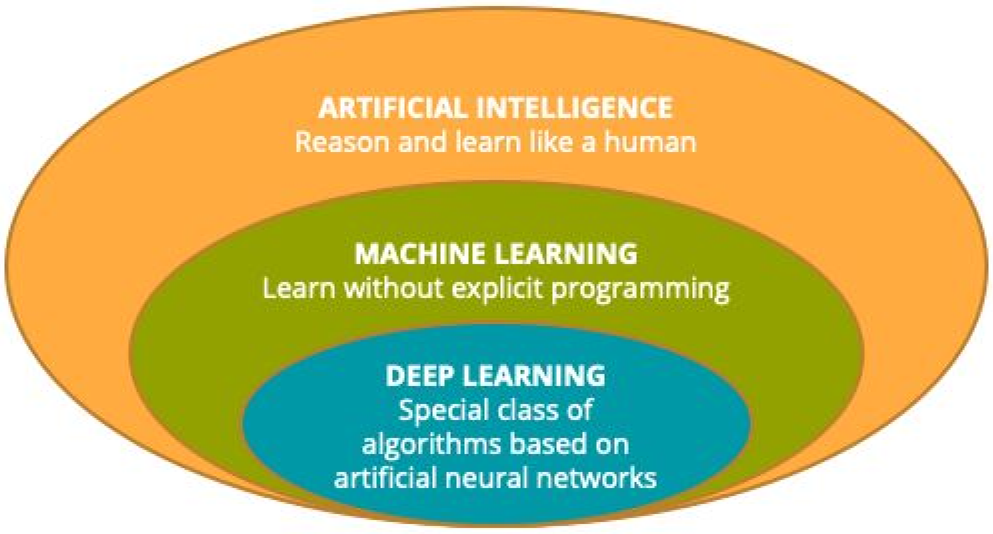
- *Similarity learning* and the basic features of a recommendation engine
- *Text classification* and the fundamentals of processing text in machine learning
- *Feature learning*, an essential task in feature engineering
- Anomaly detection
- Time-series forecasting.

Along the way, you will get practice with several hands-on labs, in which you will train a simple neural network, train a recommendation engine, train a text classifier, and get some experience with forecasting

NEXT

Classical Machine Learning vs. Deep Learning

As we just described, Artificial Intelligence (AI) *includes* Machine Learning (ML), which *includes* Deep Learning (DL). We can visualize the relationship like this:


ARTIFICIAL INTELLIGENCE
Reason and learn like a human
MACHINE LEARNING
Learn without explicit programming
DEEP LEARNING
Special class of algorithms based on artificial neural networks

As the diagram shows, all deep learning algorithms are particular cases of machine learning algorithms—but it's *not* true that all machine learning algorithms are deep learning algorithms.

A More Detailed Comparison

QUIZ QUESTION

Which of the following statements describes most accurately the relationship between classical Machine Learning and Deep Learning?

- Machine Learning algorithms are a special case of Deep Learning algorithms and have the capability of learning accurately complex, non-linear functions from data

- Deep Learning and Machine Learning algorithms are two completely different sets of algorithms
- **Deep Learning algorithms are a special case of Machine Learning algorithms and have the capability of learning accurately complex, non-linear functions from data**
- Machine Learning and Deep Learning algorithms are essentially equivalent, and can be easily translated into each other.

SUBMIT

NEXT

What is Deep Learning?

QUIZ QUESTION

Which field emerged to address the problem of *learning functions from data without explicit programming*?

- Artificial intelligence
- **Machine learning**
- Deep learning

SUBMIT

The diagram shown in the above video is from Deep Learning, by Ian Goodfellow, Yoshua Bengio, Aaron Courville. The entire book is available for free through their website. In case you'd like to dig more into the comparisons we discussed here, the diagram and accompanying discussion of deep learning can be found in the introduction here.

A Word of Caution About "Neural"

Characteristics of Deep Learning

QUIZ QUESTION

Which of the below statements are true about deep learning?

(Select all that apply.)

- **It can be very computationally expensive, especially as model complexity increases**
- It requires well-processed, highly structured data
- **It works well with massive amounts of training data**
- **It is capable of extracting features automatically**

SUBMIT

NEXT

Benefits and Applications of Deep Learning

Benefits of Deep Learning

QUIZ QUESTION

Which one of the following statements about deep learning algorithms is *not* true?

- They can be used on very large sets of data.
- They can learn complex patterns without explicitly seeing those patterns.
- They can be distributed for parallel training.
- A key benefit of deep learning algorithms is the ability to learn arbitrarily complex functions.
- **They usually use less computational resources as compared to other approaches.**

SUBMIT

Applications of Deep Learning

Now that we've reviewed some of the key characteristics, benefits, and applications of deep learning, let's get some hands-on practice building a simple neural net in Azure Machine Learning Studio.

NEXT

Specialized Cases of Model Training

In this section, we'll have a look at some of the specialized cases of model training. But first, let's review the main types of machine learning approaches that we introduced back in the first lesson.

A Review of Approaches to Machine Learning

As you now know, there are three main approaches to machine learning:

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**

QUIZ QUESTION

Each of the approaches listed below can be categorized as a type of *supervised learning*, *unsupervised learning*, or *reinforcement learning*. Can you match them up correctly?

Reinforcement learning

Unsupervised learning

APPROACH	SUPERVISED, UNSUPERVISED, OR REINFORCEMENT
Markov decision process	Reinforcement learning
Classification	Supervised learning
Similarity learning	Supervised learning
Clustering	

Unsupervised learning

SUBMIT

Specialized Cases of Model Training

NEXT

Similarity Learning

QUESTION 1 OF 2

Suppose a streaming service uses similarity learning to compare movies and generate a continuous numerical value (e.g., 0.84) indicating how similar the movies are. This would be an example of similarity learning as…

- classification
- **regression**
- ranking

SUBMIT

Recommender Systems

As we mentioned above, one of the most common uses of similarity learning is in creating *recommender systems*. Let's consider these in more detail.

QUESTION 2 OF 2

Which type of recommender system requires the least amount of information about users and items?

- Content-based recommender
- Option-based recommender
- **Collaborative filtering recommender**

SUBMIT

NEXT

Text Classification

Remember that before we can do text classification, the text first needs to be translated into some kind of numerical representation—a process known as **text embedding**. The resulting numerical representation, which is usually in the form of vectors, can then be used as an input to a wide range of classification algorithms.

Training a Classification Model with Text

As we mentioned in the video, an important part of the pipeline is the process of vectorizing the text, using a technique such as Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. We discussed text vectorization in some detail back in the introduction lesson when we described text data—so you may want to go back to that section if you feel like you need a review of the concept.

QUIZ QUESTION

Which of the following is not a typical step in model training for text classification?

- **Document labeling**
- Text normalization
- Supervised learning model training
- Feature extraction

SUBMIT

NEXT

Feature Learning

As we've discussed previously, *feature engineering* is one of the core techniques that can be used to increase the chances of success in solving machine learning problems. As a part of feature engineering, **feature learning** (also called **representation learning**) is a technique that you can use to derive new features in your dataset. Let's have a look at how this technique works.

Supervised and Unsupervised Approaches

Earlier in this lesson, we pointed out that *feature learning* is one of the machine learning techniques that can be done in both *supervised* and *unsupervised* ways. Let's have a look at both approaches.

QUIZ QUESTION

Which of the following is not a form of feature learning?

- Clustering
- Principal Component Analysis
- Unsupervised autoencoding with Deep Learning

- Supervised encoding with Deep Learning

- **Regression**

SUBMIT

Applications of Feature Learning

As we just mentioned, some prominent applications of machine learning include *image classification* and *image search*. On the remainder of this page, we'll take a closer look at some specific examples of these two approaches.

Image Classification with Convolutional Neural Networks (CNNs)

Image Search with Autoencoders

NEXT

Anomaly Detection

Datasets often contain a small number of items that deviate significantly from the norm. These *anomalies* can be of interest, since they may be the result of bad data, unusual behavior, or important exceptions to the typical trends. **Anomaly detection** is a machine learning technique concerned with finding these data points.

Supervised and Unsupervised Approaches

Anomaly detection is another one of the machine learning techniques that can be done in both *supervised* and *unsupervised* ways. Let's have a look at both approaches.

QUESTION 1 OF 2

Which of the following is true about both supervised and unsupervised approaches to anomaly detection?

- **The "anomaly" and "normal" data points are highly imbalanced**
- It is a binary classification problem
- It uses training data that has no normal/anomaly labels available
- It involves identifying two major groups (clusters) of entities

SUBMIT

Applications of Anomaly Detection

Let's now look at a specific example of anomaly detection, so that we can get a more concrete idea of what the process might look like. In this particular example, we'll consider what anomaly detection might look like when applied to *machinery maintenance.*

QUESTION 2 OF 2

Which of the following is not a typical application of anomaly detection?

- Outlier detection
- Condition monitoring
- Anti-malware protection
- **Image recognition**
- Fraud detection

SUBMIT

SUBMIT

Applications of Deep Learning

Now that we've reviewed some of the key characteristics, benefits, and applications of deep learning, let's get some hands-on practice building a simple neural net in Azure Machine Learning Studio.

NEXT

Forecasting

Now let's have a look at the problem of forecasting. A typical example of a forecasting problem would be: Given a set of ordered data points (such as sales data over a series of dates), we want to predict the next data points in the series (such as what sales will look like next week).

Remember, forecasting is a class of problems that deals with predictions in the context of orderable datasets. These orderable datasets can be time-series datasets, but they don't have to be—forecasting can be applied to other types of orderable sets as well.

Types of Forecasting Algorithms

QUIZ QUESTION

Which of the following cannot be used for time-series forecasting?

- Temporal Convolutional Network
- Multivariate regressor
- Recurrent Neural Network
- **Autoencoder Neural Network**

SUBMIT

NEXT

In this section, we'll have a look at some of the specialized cases of model training. But first, let's review the main types of machine learning approaches that we introduced back in the first lesson.

A Review of Approaches to Machine Learning

As you now know, there are three main approaches to machine learning:

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**

QUIZ QUESTION

Each of the approaches listed below can be categorized as a type of *supervised learning*, *unsupervised learning*, or *reinforcement learning*. Can you match them up correctly?

Unsupervised learning

Reinforcement learning

APPROACH

SUPERVISED, UNSUPERVISED, OR REINFORCEMENT

Markov decision process

Reinforcement learning

Classification

Supervised learning

Similarity learning

Supervised learning

Clustering

Unsupervised learning

SUBMIT

Lesson Summary

In this lesson, you've learned the fundamentals of **deep learning**, including:

- The differences between classical machine learning and deep learning
- The benefits and applications of Deep Learning
- How to train your first neural network model

Next, you learned about some of the most important **specialized cases of model training**, including:

- *Similarity learning* and the basic features of a recommendation engine
- *Text classification* and the fundamentals of processing text in machine learning
- *Feature learning*, an essential task in feature engineering
- Anomaly detection
- Time-series forecasting.

Along the way, you got practice with several hands-on labs, using the Designer in Azure Machine Learning Studio to train a simple neural network, a recommendation engine, a text classifier, and a time-series forecasting model.

Lesson Overview

This lesson covers **managed services for Machine Learning**, which are services you use to enhance your Machine Learning processes. We will use services provided by Azure Machine Learning as examples throughout the lesson.

You will learn about various types of **computing resources** made available through managed services, including:

- Training compute
- Inferencing compute
- Notebook environments

You will also study the main concepts involved in the **modeling process**, including:

- Basic modeling

- How parts of the modeling process interact when used together
- More advanced aspects of the modeling process, like automation via pipelines and end-to-end integrated processes (also known as DevOps for Machine Learning or simply, MLOps)
- How to move the results of your modeling work to production environments and make them operational

Finally, you will be introduced to the world of programming the managed services via the **Azure Machine Learning SDK for Python**.

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 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.

NEXT

Compute Resources

A compute target is a designated compute resource or environment where you run training scripts or host your service deployment. There are two different variations on compute targets that we will discuss below: training compute targets and inferencing compute targets.

Training Compute

First, let's talk about compute resources that can be used for model training.

Inferencing Compute

Once you have a trained model, you'll want to be able to deploy it for inferencing. Let's take a look at the compute resources you can use for different types of inferencing.

QUIZ QUESTION

Which of the following compute targets are *not* managed by Azure Machine Learning?

- Compute instance
- Azure Container Instances
- Compute clusters

SUBMIT

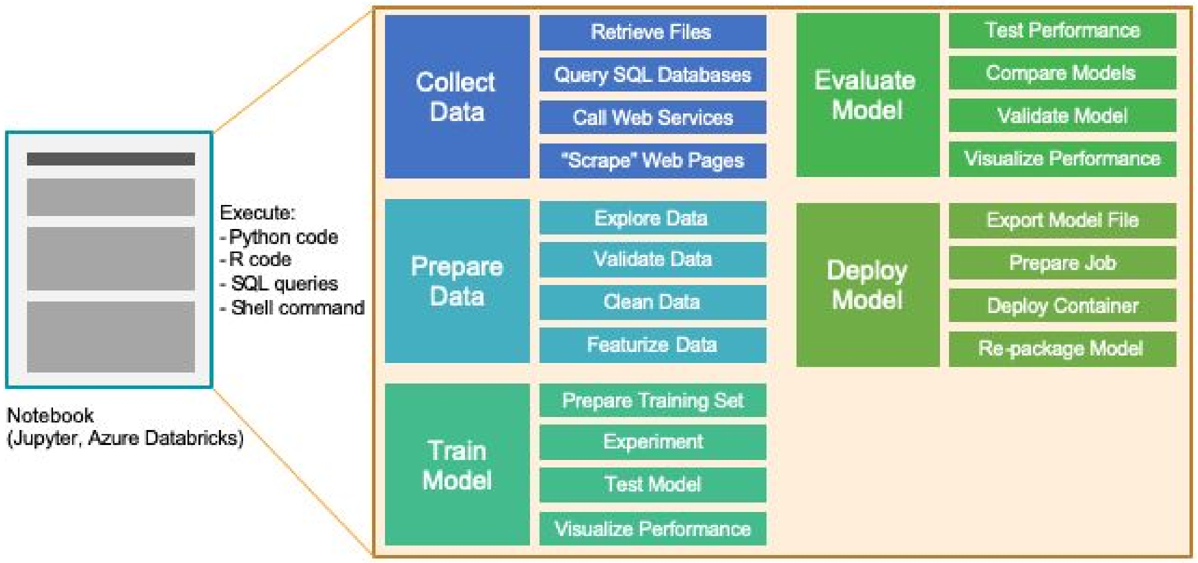
NEXT

correct!

Azure Container Instances are not managed by Azure Machine Learning. A managed compute resource is created and managed by Azure Machine Learning. This type of compute is optimized for machine learning workloads. Azure Machine Learning compute clusters and compute instances are the only managed computes.

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:



The best way to understand how managed notebook environments work is to jump in and work with one—so that's what we'll do in the next lab, by training a simple SciKit learn model on a medical dataset.

NEXT

Basic Modeling

Training, evaluating, and selecting the right Machine Learning models is at the core of each modern data science process. But what concrete steps do we need to go through to produce a trained model? In this section, we'll look at some important parts of the process and how we can use Azure Machine Learning to carry them out.

Experiments

Before you create a new *run*, you must first create an experiment. Remember, 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. As we discussed above, model training runs are what you use to build the trained model. 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. To get a model, we start with a more general algorithm. By combining this algorithm with the training data—as well as by tuning the hyperparameters—we produce a more specific function that is optimized for the particular task we need to do. Put concisely:

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. Note that models are either produced by a Run or originate from outside of Azure Machine Learning (and are made available via model registration).

QUIZ QUESTION

When you train a model in Azure Machine Learning, there are a few steps you must perform, in a particular order. Which of these shows the correct order?

- Create a new run in code, create a new experiment after the run is completed, then register the trained model.
- Start out by creating a new registry for the model. Next, create a new experiment and add a run to it. The model is automatically associated to its registration.
- **Create a new experiment for your runs. Submit a script run. After the run successfully completes, register the model.**

SUBMIT

NEXT

Correct!

An **experiment** is a generic context for handling and organizing runs. Once you have an experiment, you'll want to create one or more **runs** within it, until you identify the best model. You then register the final model in a **model registry**. After registration, you can then download or deploy the registered model and receive all the files that were registered.

Advanced Modeling

Machine Learning Pipelines

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, there can be many steps involved in the end-to-end process, including:

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

These steps are organized into *machine learning pipelines*.

MLOps: Creating Automatic End-to-End Integrated Processes

As we said earlier, we don't want all the steps in the machine learning pipeline to be manual—rather, we want to develop processes that use automated builds and deployments. The general term for this approach is *DevOps*; when applied to machine learning, we refer to the automation of machine learning pipelines as **MLOps**.

NEXT

Operationalizing Models

After you have trained your machine learning model and evaluated it to the point where you are ready to use it outside your own development or test environment, you need to deploy it somewhere. Another term for this is **operationalization**.

Real-time Inferencing

The model training process can be very compute-intensive, with training times that can potentially spann 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.**

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.

QUIZ QUESTION

When you deploy a Machine Learning model used for real-time scoring, or batch inferencing, there are several steps you must follow. Select the *required* steps in the list below.

(Select all that apply.)

- **Save and retrieve the model file in any format**
- Create a metadata file that describes the model

- Create a training script
- **Create a scoring script**
- Create a schema file that describes the web service input
- **Create a real-time scoring web service**

SUBMIT

NEXT

Programmatically Accessing Managed Services

Azure Machine Learning provides a code-first experience via the Azure Machine Learning SDK for Python. Using the SDK, you can start training your models on your local machine and then scale out to use Azure Machine Learning compute resources. This allows you to train better performing, highly accurate machine learning models.

Azure Machine Learning service supports many of the popular open-source machine learning and deep learning Python packages that we discussed earlier in the course, such as:

- Scikit-learn
- Tensorflow

- PyTorch
- Keras

In the next lab, we'll get some practice using Azure ML Python SDK to register, package, and deploy a trained model.

Lesson Summary

In this lesson you've learned about **managed services for Machine Learning** and how these services are used to enhance Machine Learning processes.

First, you learned about various types of **computing resources** made available through managed services, including:

- Training compute
- Inferencing compute
- Notebook environments

Next, you studied the main concepts involved in the **modeling process**, including:

- Basic modeling
- How parts of the modeling process interact when used together
- More advanced aspects of the modeling process, like automation via pipelines and end-to-end integrated processes (also known as DevOps for Machine Learning or simply, MLOps)
- How to move the results of your modeling work to production environments and make them operational

Finally, you were introduced to the world of programming the managed services via the **Azure Machine Learning SDK for Python**.

NEXT

Lesson Overview

This lesson will introduce you to the essential and difficult discussion about the potential implications and challenges posed by Machine Learning. In this lesson, we will explore:

- The modern-day challenges posed by AI in general and Machine Learning in particular.
- The core principles of responsible AI (as a broader perspective of Machine Learning).

- How Microsoft applies these principles.
- Two essential aspects of Machine Learning models that impact responsible AI: *transparency* and *explainability*.

Modern AI: Challenges and Principles

Modern AI: Challenges and Principles

QUIZ QUESTION

Below are some of the examples we just discussed. Can you match each with the type of problem it exemplifies?

Submit to check your answer choices!

THIS IS AN EXAMPLE...

...OF THIS PROBLEM:

In the US alone, it is estimated there will be a shortage of 250,000 data scientists by 2025.

Skilled labor deficit

An advertising system accidentally showed an ad for high-income jobs to men more often than it showed the ad to women.

Unintentional bias

Your self driving car is fooled by someone who carefully applied a specially designed sticker to a stop sign.

Adversarial attacks

A video shows a world leader appearing to say something they never actually said.

Deep fakes

Someone deliberately manipulates the public data used for model training.

Intentional data poisoning

An infinitesimally small change in the input to a model causes huge, poorly understood impact.

The butterfly effect

SUBMIT

Lesson 7:

Responsible AI

SEARCH
RESOURCES
CONCEPTS

Microsoft AI Principles

SEND FEEDBACK

Microsoft AI Principles

QUESTION 1 OF 2

Below are the Microsoft AI principles. Can you match each one with the correct description?

Submit to check your answer choices!

DESCRIPTION

PRINCIPLE

AI systems should empower everyone and engage people

Inclusiveness

AI systems should be understandable

Transparency

Algorithms and the people who write them should be responsible or answerable for their impacts.

Accountability

AI systems should treat all people fairly

Fairness

AI systems should perform consistently and minimize risk.

Reliability & Safety

AI systems should protect people and their personal data.

Privacy and Security

SUBMIT

QUESTION 2 OF 2

Which two principles are foundational principles that ensure the effectiveness of the other principles?

(Select exactly two options.)

- Privacy and Security
- **Accountability**
- Reliability and Safety
- Fairness
- Inclusiveness
- **Transparency**

SUBMIT

NEXT

NEXT

Model Transparency and Explainability

How to Understand and Explain Models

QUIZ QUESTION

Categorize the following items as relating to **direct explainers** or **meta explainers**.

Submit to check your answer choices!

CONCEPT	DIRECT OR META EXPLAINER
Helps with explainer selection	**Meta Explainer**
Model agnostic explainer	**Direct Explainer**
SHAP Tree Explainer	**Direct Explainer**
Text Explainer	**Meta Explainer**

SUBMIT

NEXT

Model Fairness

In this next video, we'll walk through an example of how you could use some of the Fairlearn capabilities when applying model fairness to your work. This demo is just intended as an example, not a lab exercise, although you are certainly welcome to check out the FairLearn repository for yourself, including the notebooks we use in the demo, here.

Fairlearn Notebook Demo

Lesson Summary

In this lesson you've learned about the potential implications and difficult challenges posed by Machine Learning.

You've studied the core principles of responsible AI and how Microsoft aligns its AI strategy according to them. You've also learned about model transparency and explainability, and you've performed an exploration of model explanations using Azure Machine Learning Studio.

Finally, you've learned about some of the modern-day challenges posed by AI and Machine Learning.

Course Conclusion

The Future and Importance of Machine Learning

- Machine Learning is already having a significant impact on almost every aspect of our daily lives
- Machine Learning promises to help advance critically important fields like medicine, transportation, space travel, agriculture, cybersecurity, and many more
- The trends in computing power availability, moving closer to silicon, public cloud HW resource concentration will continue to accelerate in the next decade
- Other medium or long-term scientific breakthroughs (think quantum computing) have the potential of creating even larger ML advancements
- Better understanding of current algorithms, inventing new ones, focusing on transparency and explainability will be also major trends

Course recap

This course was meant to be an introduction to the fantastic world of Machine Learning. Here is a short recap of our journey:

In Lesson 1, you've learned the fundamentals of Machine Learning, made your first contact with data, models, and algorithms, and trained your first model in Azure Machine Learning Studio.

In Lesson 2, you performed a few basic data preparation tasks like import, transformation, and feature engineering. You've also trained a classifier and a regressor, and learned to apply the "strength in numbers" principle using ensembles and automated Machine Learning.

In Lesson 3, you've learned about supervised and unsupervised learning, two of Machine Learning's fundamental approaches. You've also trained and evaluated the performance of classifiers, regressors, and clustering models.

In Lesson 4, you got your first taste of Deep Learning and dived into some of the most important specialized cases like similarity learning, text classification, feature learning, anomaly detection, and forecasting.

In Lesson 5, you've learned to manage compute resources, model training and evaluation processed, and operational environment. You've also experienced the use of the Azure Machine Learning SDK for Python to program the Azure Machine Learning managed services.

Finally, in Lesson 6, you've learned the core principles of responsible AI. You've also learned to train transparent and explainable models, and raised your awareness on some of the serious challenges of modern AI and Machine Learning.

NEXT