Table of Contents

Lesson 2: Introduction to Machine Learning

Chapter 1: Overview

In this lesson, our goal is 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.

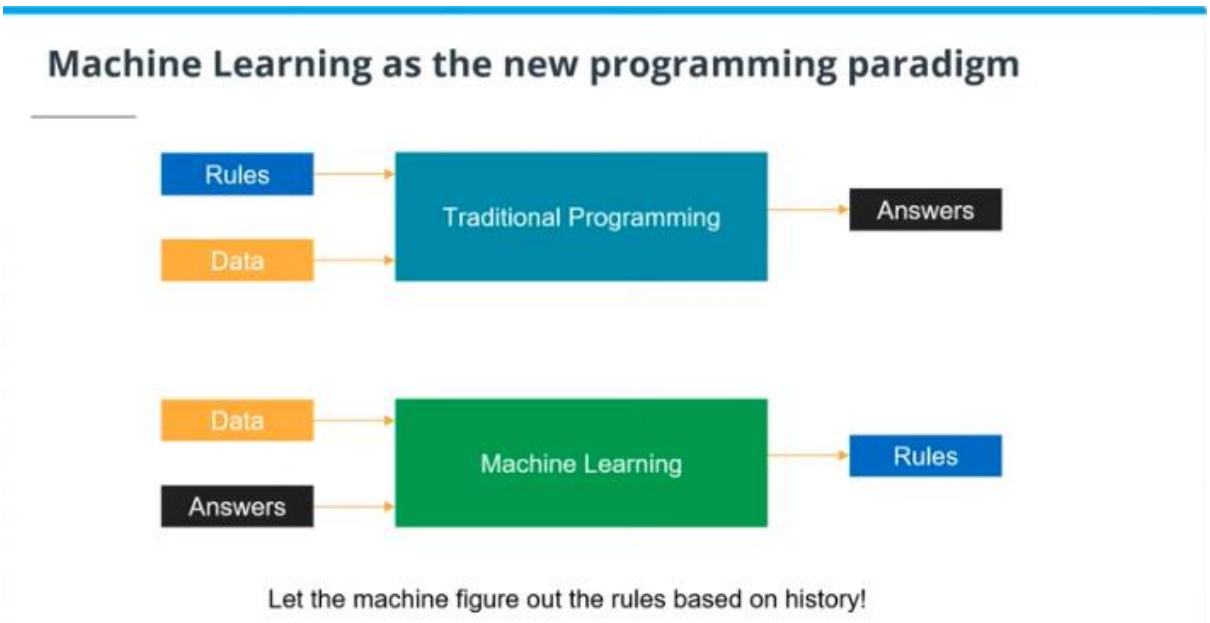
Here are the main topics we will cover:

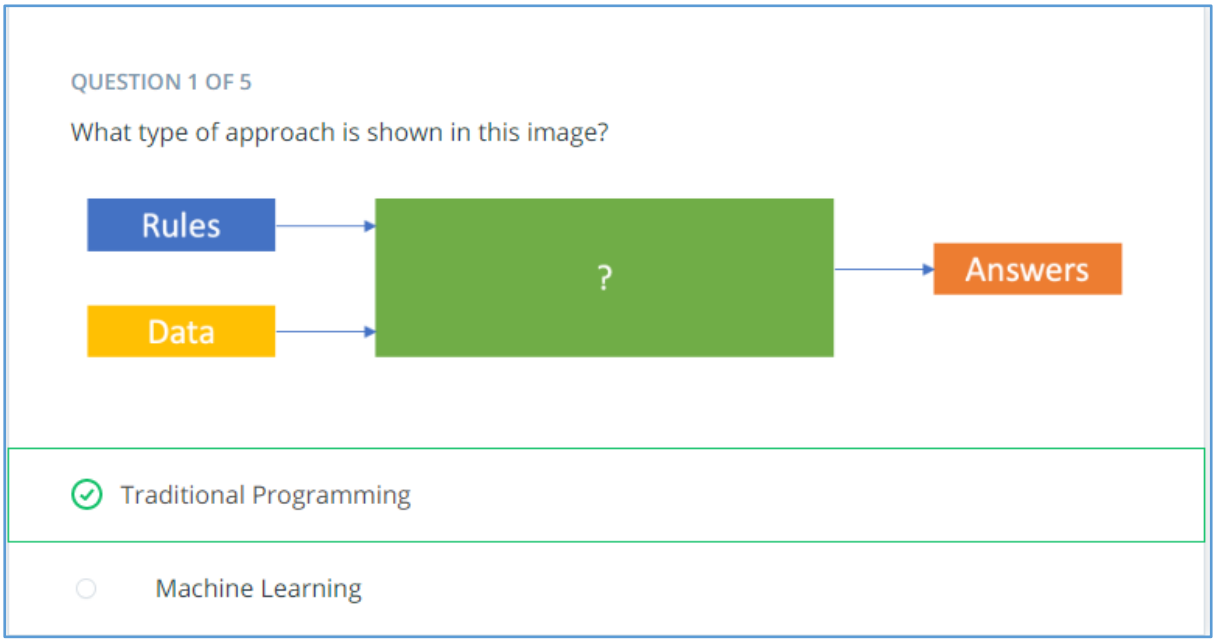
- What machine learning is and why it's so important in today's world
- The historical context of machine learning
- The data science process
- The types of data that machine learning deals with
- The two main perspectives in ML: the ***statistical* perspective** and the ***computer science* perspective**
- 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 will also train your first machine-learning model using Azure Machine Learning Studio.

Chapter 2: What is Machine Learning?

Extract patterns from Data >> Identify related data >> Forecast future outcomes, behaviours and trends





Hard coded rules and Data fed in to get the Answers as Output. Once we fed data, we need to find out set of Rules as how to use data in order to find the relevant answers.

QUESTION 2 OF 5

What type of approach is shown in this image?

Traditional Programming

Machine Learning

Data and Answers fed in to get the Rules as Output. Here we have historical data and answers fed into the system. The algorithm establishes the relationship between the two in order to derive the set of rules.

QUESTION 3 OF 5

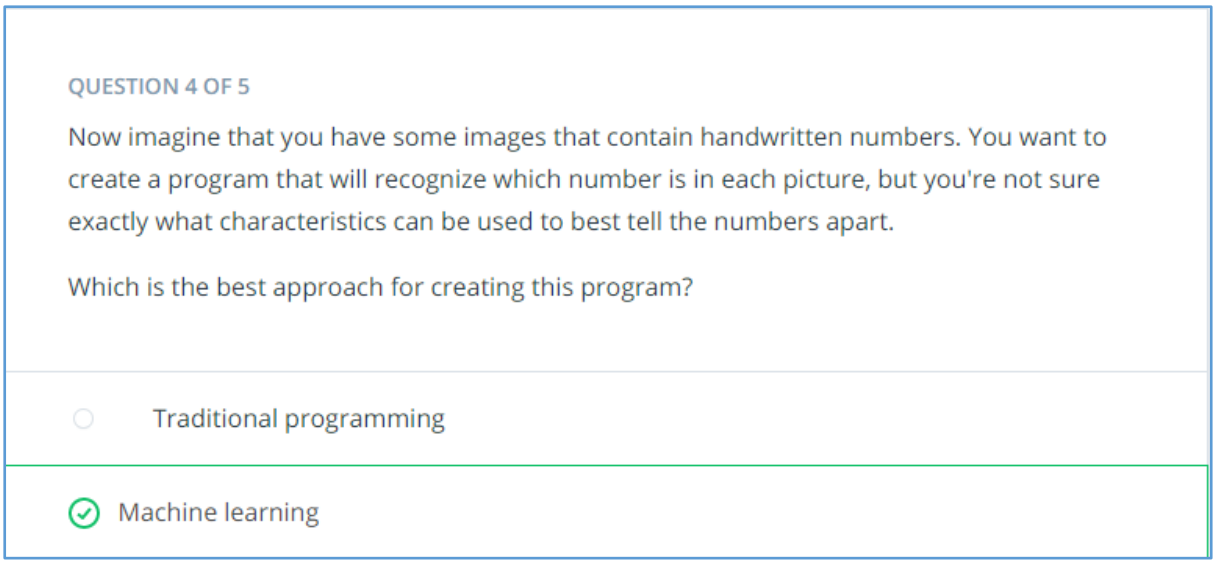
Imagine you want to create a function that multiplies two numbers together (e.g., given the inputs 2 and 3, the function will generate the output 6).

What approach is best suited to this problem?

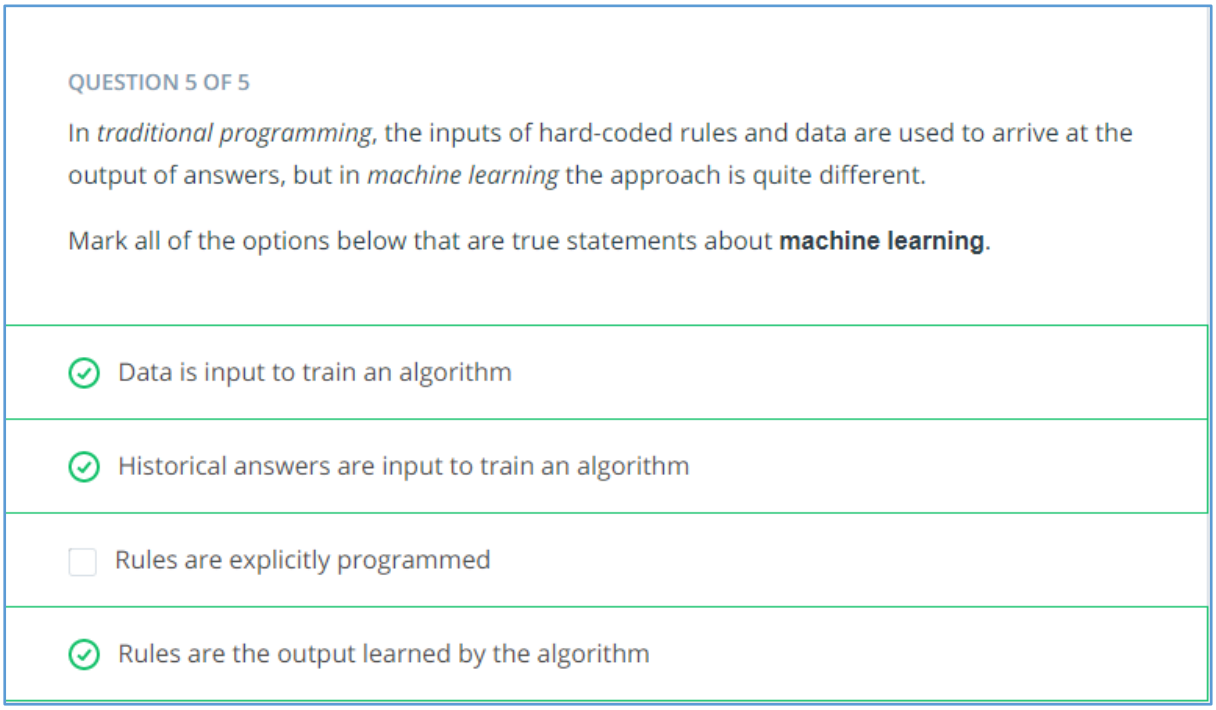
Traditional Programming

Machine Learning

Traditional programming is well suited to a problem like this, in which you are given, the data (two numbers) and already know the rules (multiplication) for getting the desired output.



Machine learning is well suited to problems like this, in which you have the data (images) and the answers (you know which images have which numbers), but it is unclear what the rules are.



Chapter 3: Applications of Machine Learning

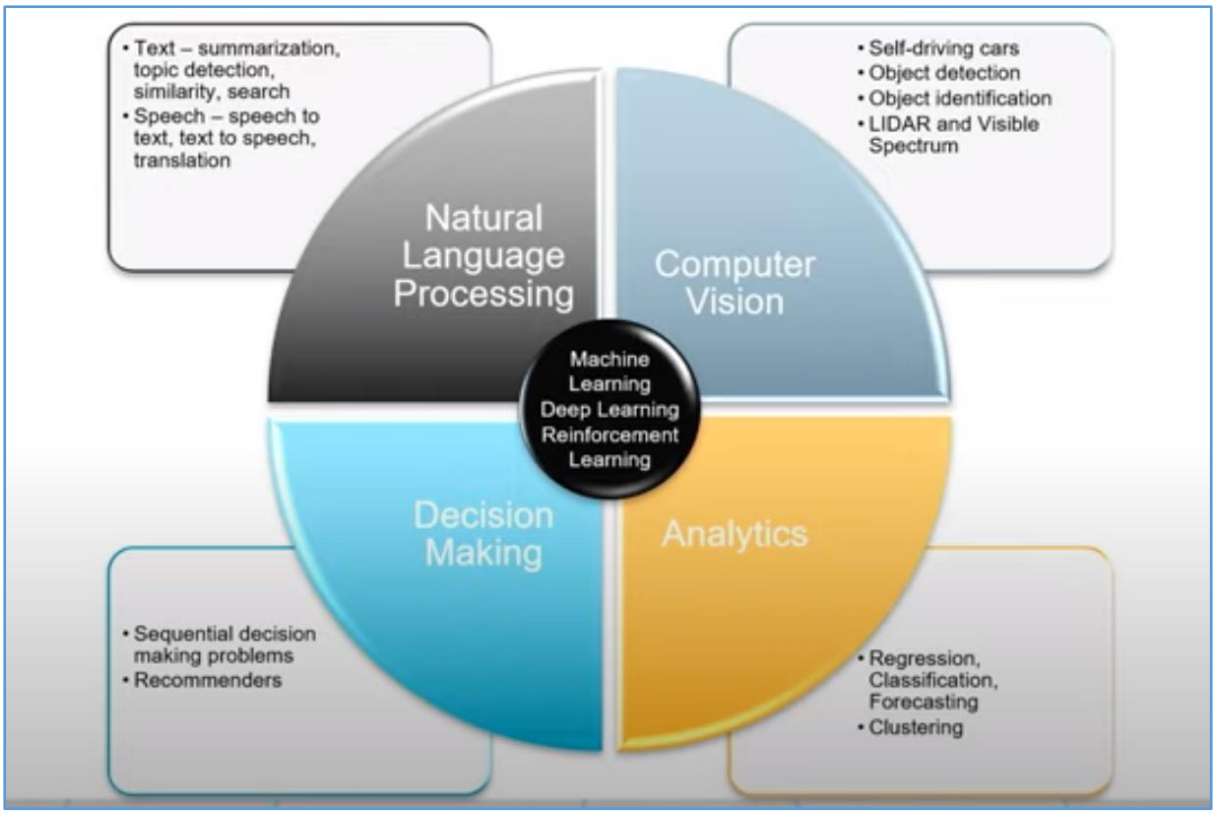
The applications of machine learning are extremely broad! & the opportunities cut across industry verticals. Whether the industry is healthcare, finance, manufacturing, retail, government, or education, there is enormous potential to apply machine learning to solve problems in more efficient and impactful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

Applications used frequently are **Statistical ML, Deep Learning and Reinforcement Learning**

- **ML is very useful in dealing with NLP applications dealing with Text and Speech. It can be used for:**

a) Summarising Texts
b) Power search – Automatically detect topics and identify if two texts are similar.
c) Speech to Texts, converting recorded human speech to texts or texts to spoken audio.

- **ML can also power Computer Vision application**
  a) Self-driving cars
  b) Object detection – Describing contents of an image
  c) Object Identification – Locating objects within an image
  d) Images created in visible spectrum that we can see, also other image types like LIDAR images that are collected by laser scanners
- **ML can also power Analytics application** that helps us making prediction performing tasks like regression, Classification, Forecasting and Clustering
- **ML can also power Decision making applications** like sequential decision making problems (Playing video games) or optimizing content recommendations based on user interactions.



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 – Computer Vision

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. Enables evaluation of higher volume of patient data.

Recommend next best actions for individual care plans – Decision Making

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.

Enable personalized, real-time banking experiences with chatbots [1st wave of customer contact] – Natural Language Processing

You have 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.

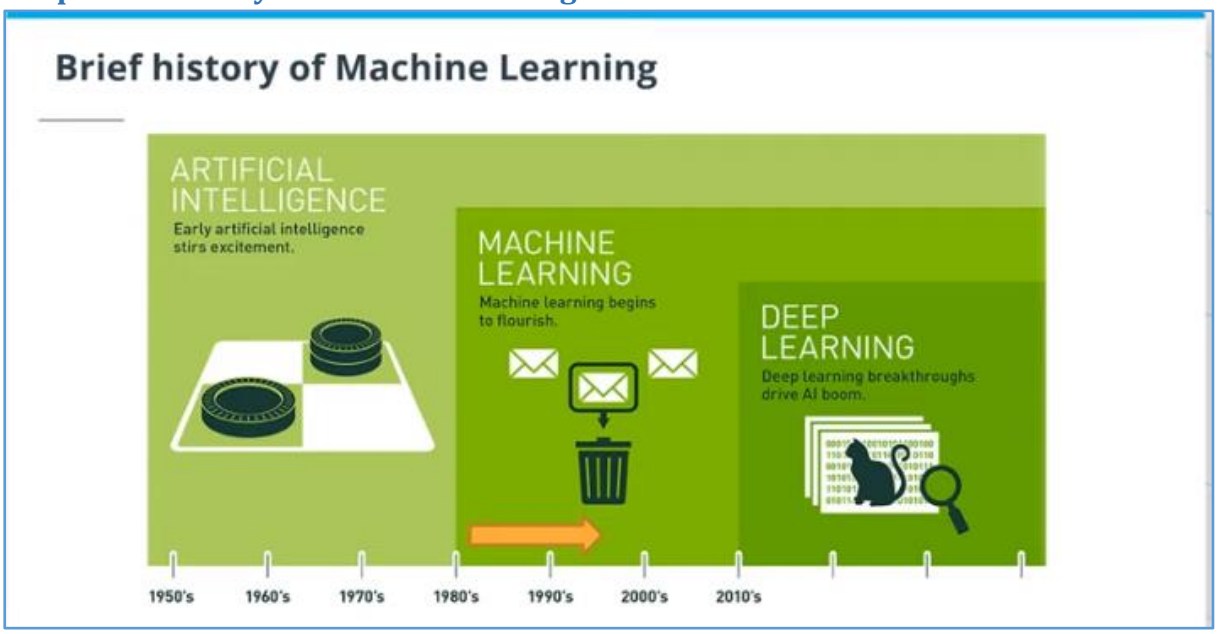
Identify the next best action for the customer – Decision Making

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. Enhanced opportunities for Cross selling and up selling with higher likelihood of acceptance.

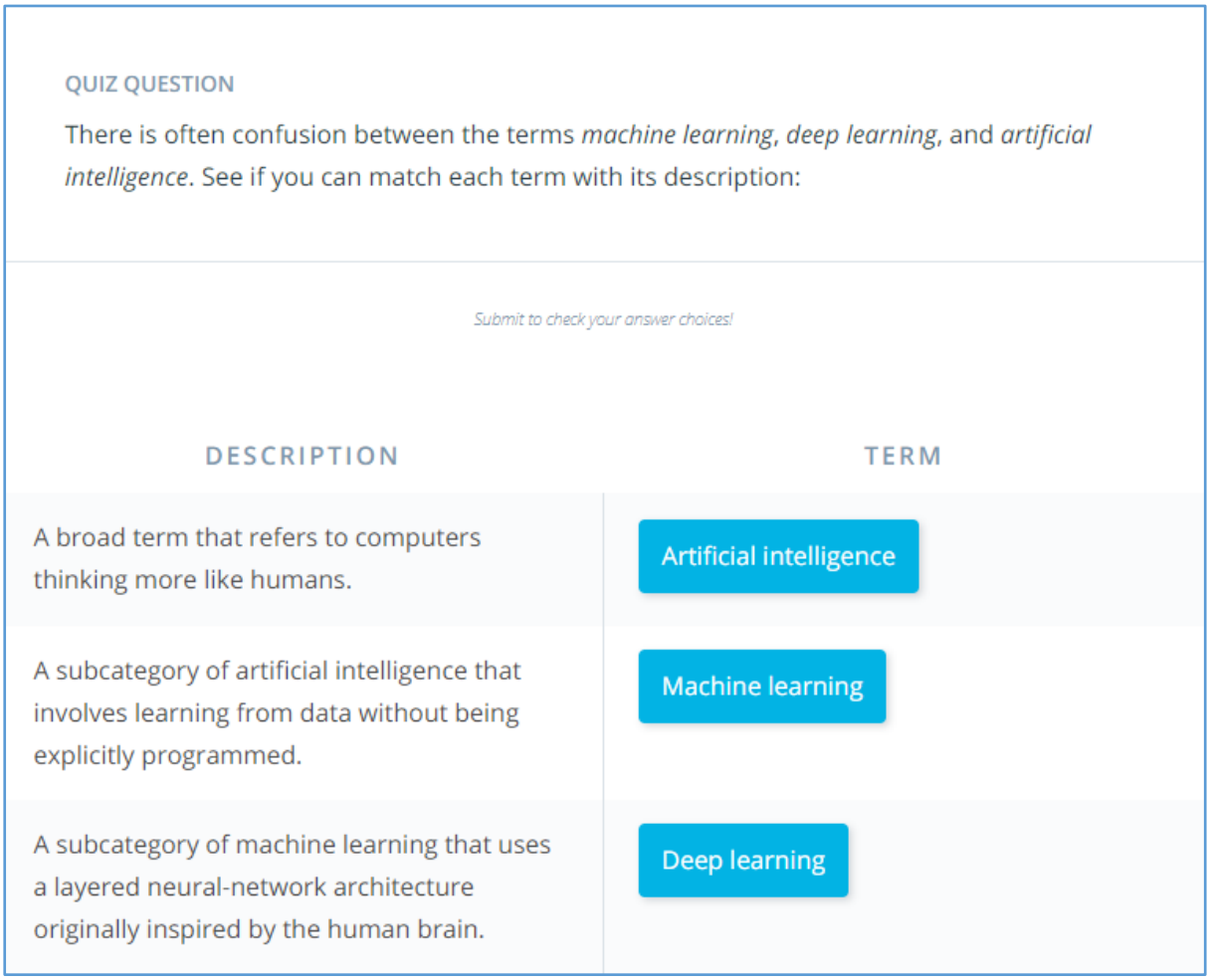
Capture, prioritize, and route, service requests to the correct employee, and improve response times - Analytics

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.

Chapter 4: History of Machine Learning



- **1950's - AI came up where the main motive was building programs that was copying the way humans think and act.**
- **1980's - Focus shifted towards specific areas of AI. One of the area is ML, which is writing programs to identify patterns without explicitly being, programed to do so. Neural nets were, inspired by the structure of human brains.**
- **2010's - Power of computers got neural nets back into spotlight. Large neural nets with many hidden layers and hidden nodes. Examples are Language processing and Image classification. That is how the term Deep Learning came into picture.**

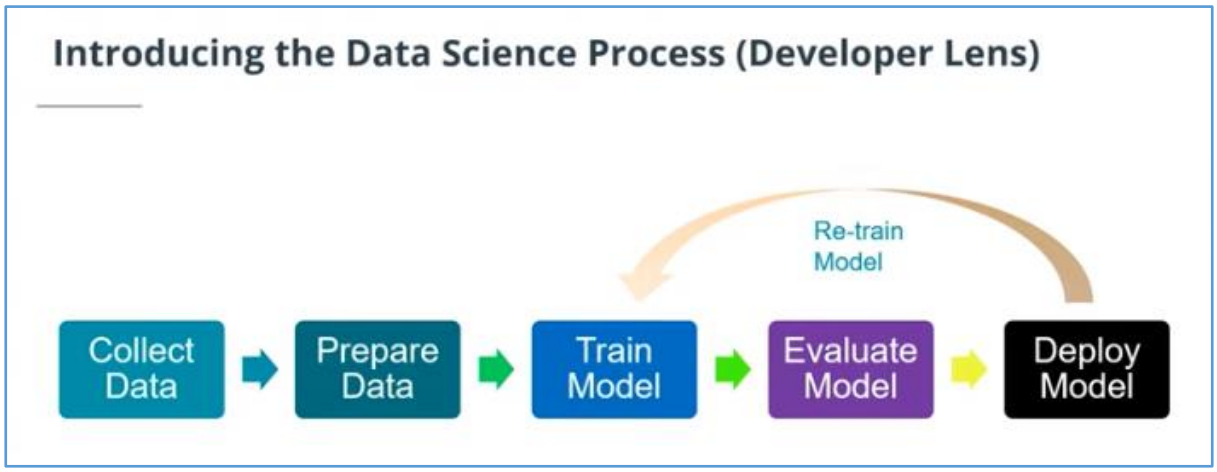


Chapter 5: Data Science Process

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 modelling 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.

- **Collect Data** – Write codes to retrieve files, Query DBs, calling Web services, Scrapping web pages.

- **Prepare Data** – Data wrangling to get the data in a format, which is suitable for analysis also we need to explore and understand data. Identify and create features required for the model. **Write queries and codes to clean the data, explore and visualize the data in order to identify the features that could be, used in Model training.**
- **Train Data** – We select an Algorithm and prepare Training, Testing and Validation dataset. Then iteratively evaluate the model to identify the best performing version and sanity check the outcome. **Write code and do some math. Prepare Model-training pipeline that includes Feature vectorization, Feature Scaling and tuning the ML Algorithm. They look into model performance on validation dataset.**
- **Evaluate Model** – Final evaluation performed with the data from Validation dataset & see how it performs. **Write code and do some math. Test and compare various models by computing evaluation metrics/graphs on test data.**
- **Deploy Model** – Once happy we deploy the model. Measure the ongoing performance of the model. Data keeps on changing hence we need to Re-train the model on Fresh data. Re training is an iterative step for models in productions. **Part of DevOps that integrates training, evaluation and deployment scripts in respective builds and release pipelines. All models and deployments should be versioned and artifacts need to be, archived.**



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

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

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

Chapter 6: Common types of data

It is all Numerical in the End. Note that although we have described numerical data as a distinct category, it is actually involved in some way with all of the data types we have 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.

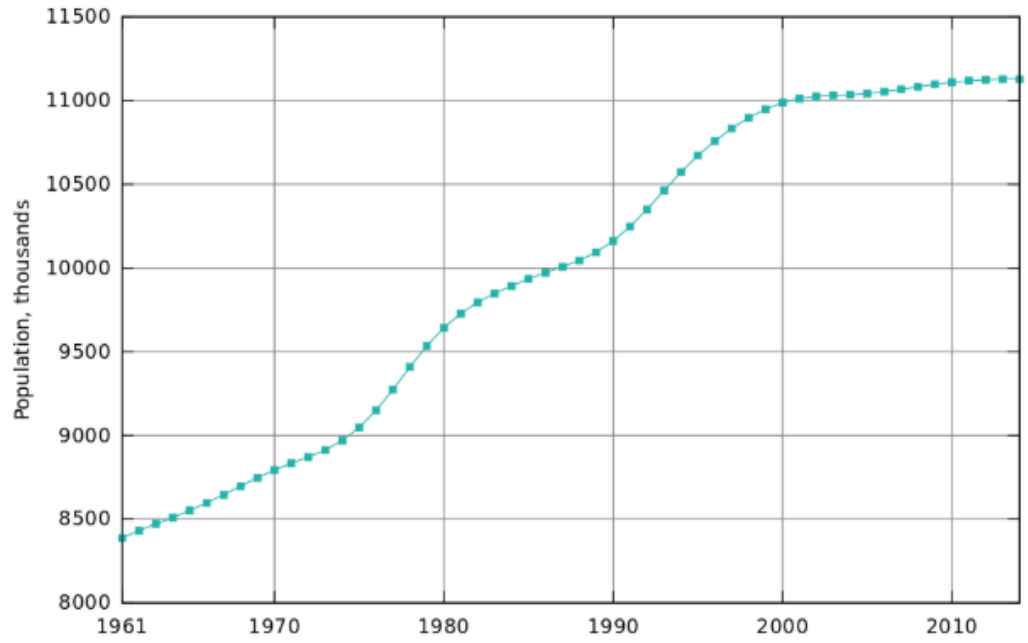
Another example would be using pictures uploaded by customers to identify if they are satisfied with the service. Pictures are not initially in numerical form but they need to be, transformed into RGB values, a set of numerical values ranging from 0 to 255, to be processed.

Data Types:

- **Numerical Data** – Integers or Floats
- **Time Series Data** – Numerical data over equally spaced time
- **Categorical Data**
- **Text Data** – Words or Sentences
- **Image Data** – Picture or Snapshot of video

QUESTION 1 OF 2

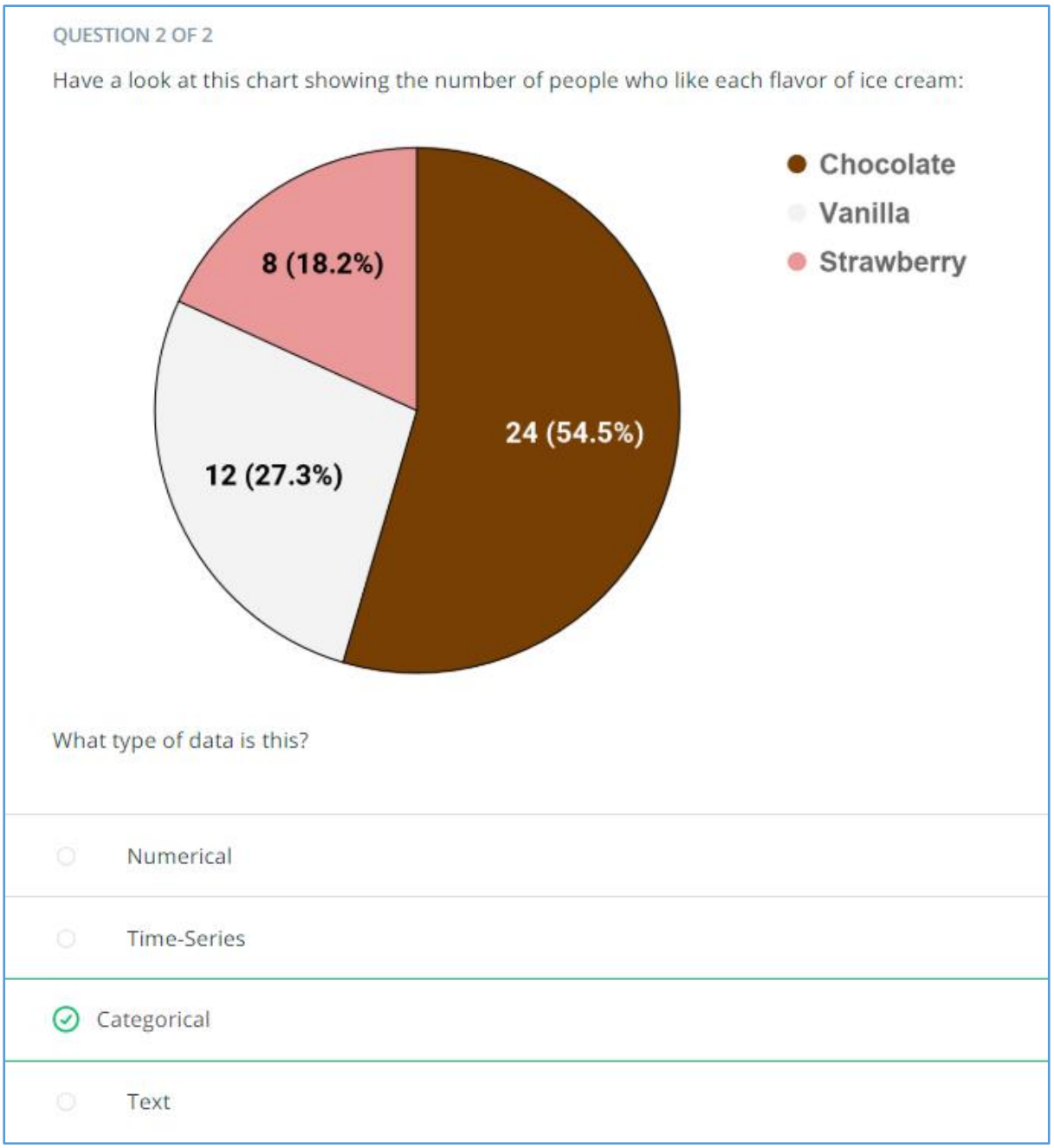
Have a look at this graph:



Population of Greece since 1961 (*Wikimedia Commons*)

What type of data is this?

- Numerical
- ✓ Time-Series
- Categorical
- Text



Chapter 7: Tabular Data

In machine learning, the most common type of data you will 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 is an example of tabular data showing some different clothing products and their properties:

SKU	Make	Colour	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

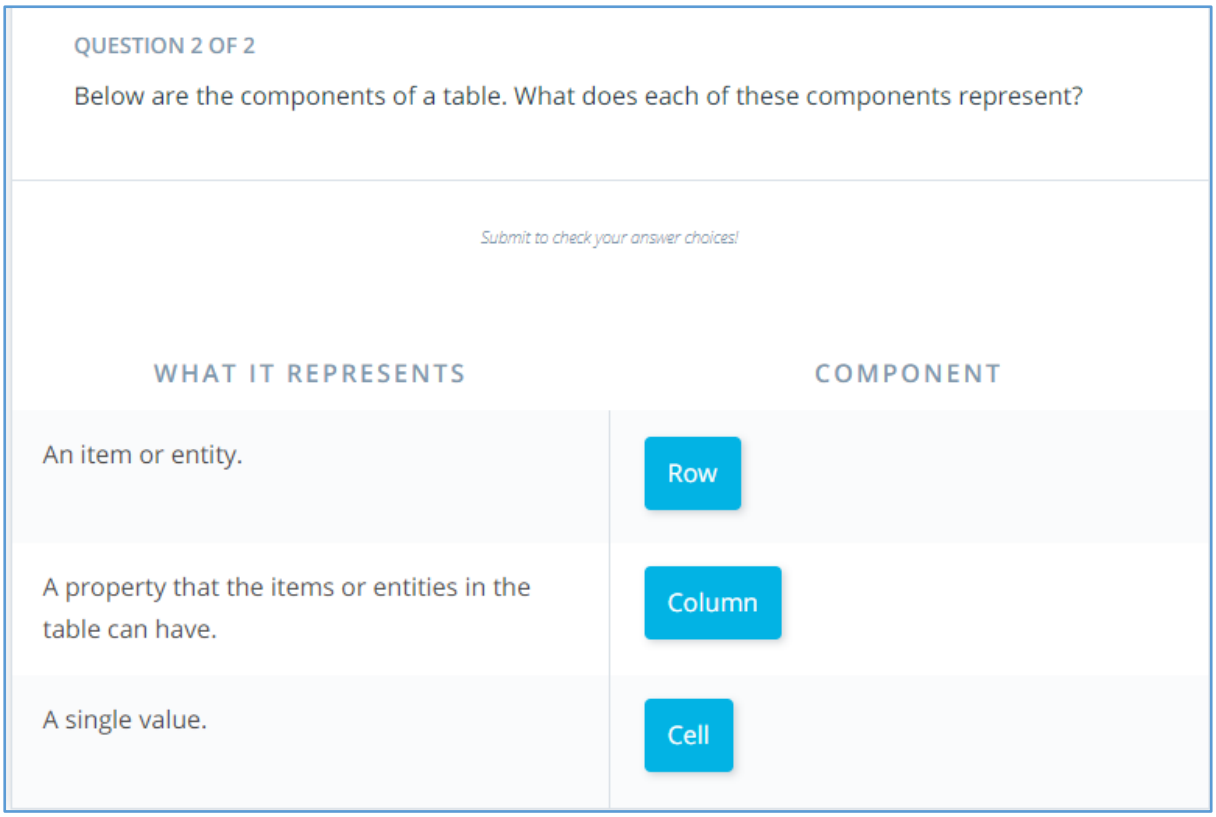
Notice how tabular data is, arranged in **rows** and **columns**.

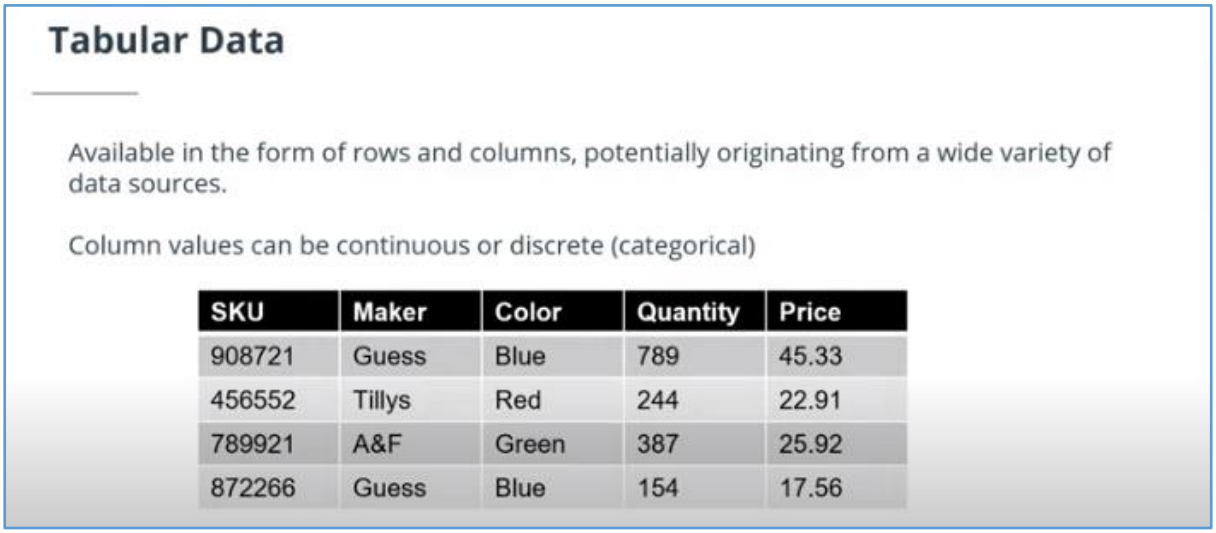
In tabular data, typically each row describes a single item, while each column describes different properties of the item.

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)





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.

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 will look at some different ways to take non-numerical data and ***vectorize*** it (that is, transform it into vector form).

Chapter 8: 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 us 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 us 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 us 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?
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

Standardize -5,10,15. Knowing that the mean is 7 and the standard deviation is 10. Use commas to separate numbers and keep one decimal place (e.g. 1.0,2.3,3.0 or -1.5,1.1,2.4)

(-1.2,0.3,0.8)

RESET

Normalize -5,10,15. Knowing that the mean is 7 and the standard deviation is 10. Use commas to separate numbers and keep one decimal place (e.g. 1.0,2.3,3.0 or -1.5,1.1,2.4)

(0.0,0.8,1.0)

RESET

Chapter 9: Encoding Categorical Data

As we have 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 us look again at our example table of clothing products:

SKU	Make	Colour	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

Moreover, if we apply it to the Colour property, we get:

Colour	Encoding
Red	0
Green	1
Blue	2

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

SKU	Make	Colour	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 Colour 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 colour 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

QUESTION 2 OF 4

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

- ○ Mammal
- ○ Reptile
- ✓ Fish

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

ID	Mammal	Reptile	Fish	Amphibian	Bird
012	1	0	0	0	0
204	0	0	1	0	0
009	0	1	0	0	0
105	1	0	0	0	0
303	0	0	0	0	1

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

Chapter 10: Image Data

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

Taking a Closer Look at Image Data

Let us 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 **coloured 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

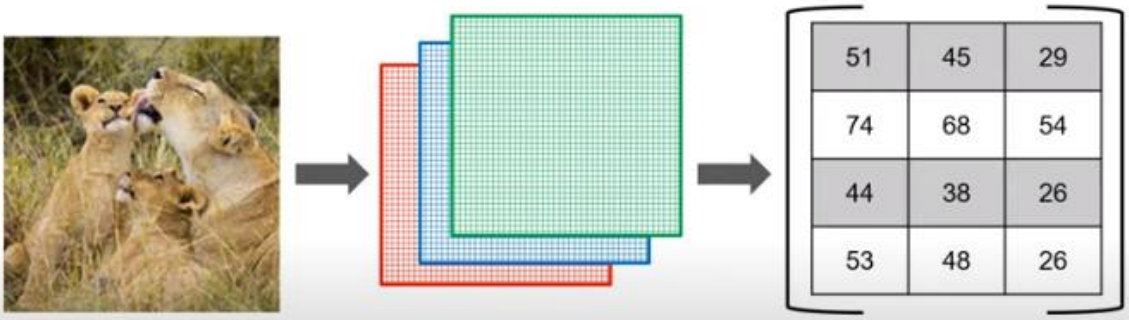
Let us now talk about how we can use this data to encode an image. We need to know the following three things about an image to reproduce it:

- 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. Total number of pixels = `height * width`

Note:

- **Depth means Channel values.**
- **In ML we use uniform aspect ratio for all images, most common being square images.**
- **Image data is, normalized to subtract per channel mean pixel values.**

Introducing Image Data



QUESTION 1 OF 2

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

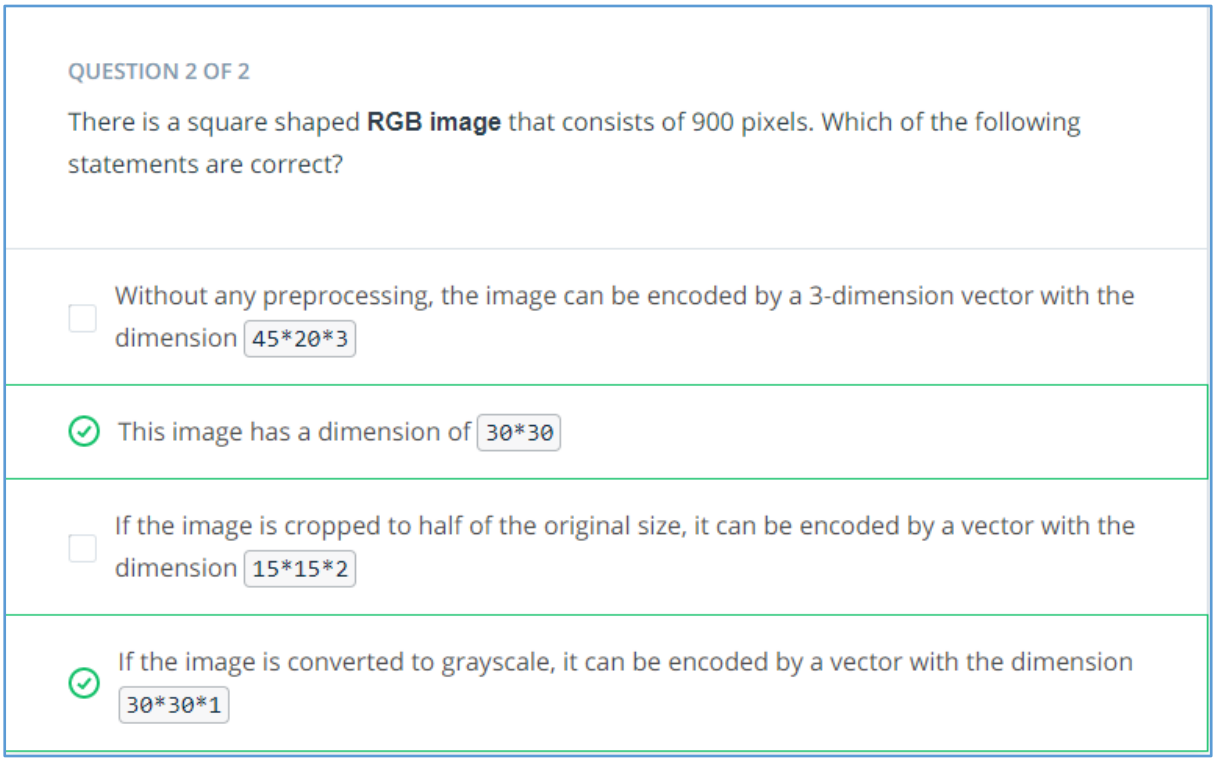
25	5	110	34
71	207	48	99
18	156	60	7
55	39	170	32

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.

Encoding with a 3-dimensional vector having the dimensions 45*20*3 would not work. Remember, this image has a *square* shape, so the height should equal the width.
We can calculate the dimension of the image by calculating the height and width. The total number of pixels (900) equals the multiplication of height and width. Moreover, since we are, told the image is *square*, this means the height and width of this image are equal.



Other Pre-processing Steps

In addition to encoding an image numerically, we may also need to do some other pre-processing 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 pre-processing operations we might want to do to clean the input images include rotation, cropping, resizing, denoising, and centering the image.

Chapter 11: 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 us 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, 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. Alternatively, 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, is, am, are example we mentioned above:

Original word	Lemmatized word
is	be
are	be
am	be

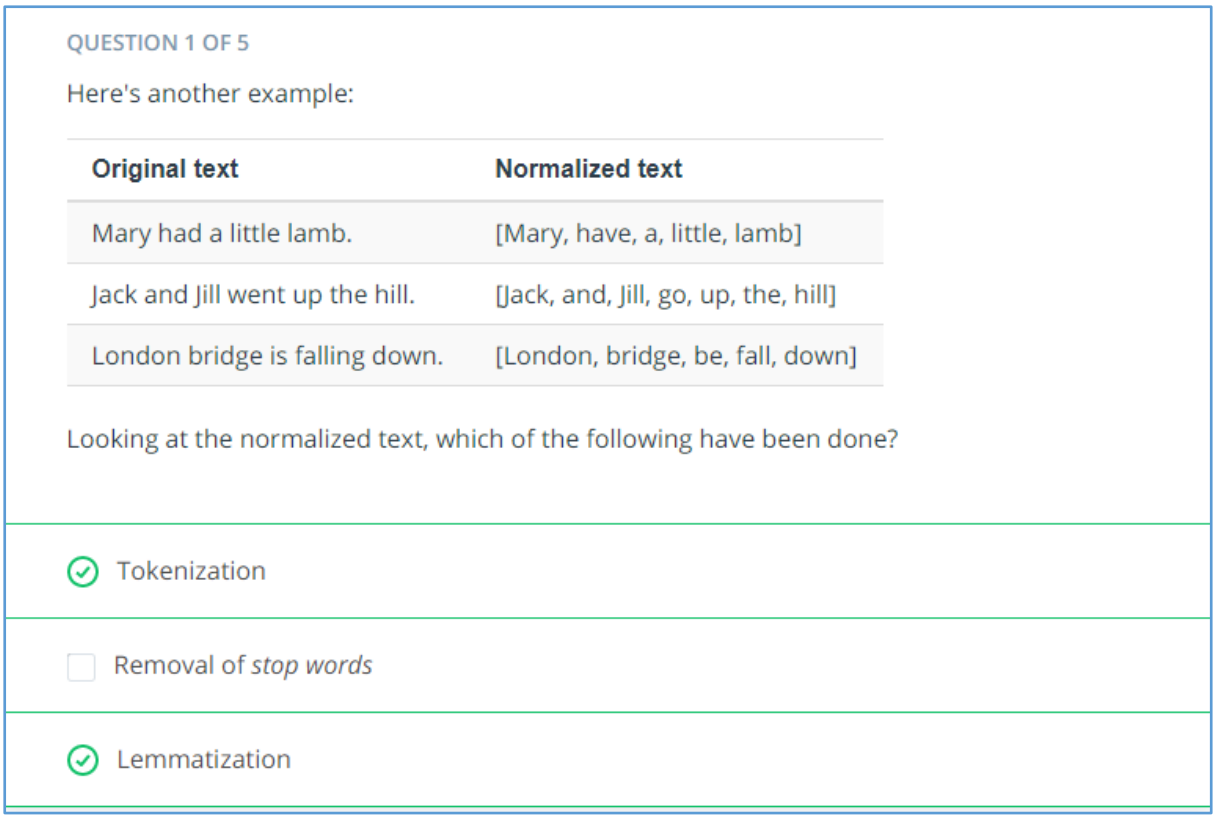
In many cases, you may also want to remove *stop words*. **Stop words** are high-frequency words that are unnecessary (or unwanted) during the analysis.

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 is 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).



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 us 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 is 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 is 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]`.

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)

If two words have a similar meaning (or some other relevant connection), the vectors used to encode them should be close to each other.

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"

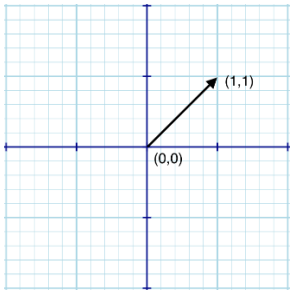
Feature Extraction

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. However, 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].

The Whole Pipeline

In this next video, we will 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 analyse 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.

Documents	Normalized Text	Vectorized Text

The quick fox.
The lazy dog.
The rabid hare.

→ [quick, fox]
[lazy, dog]
[rabid, hare]

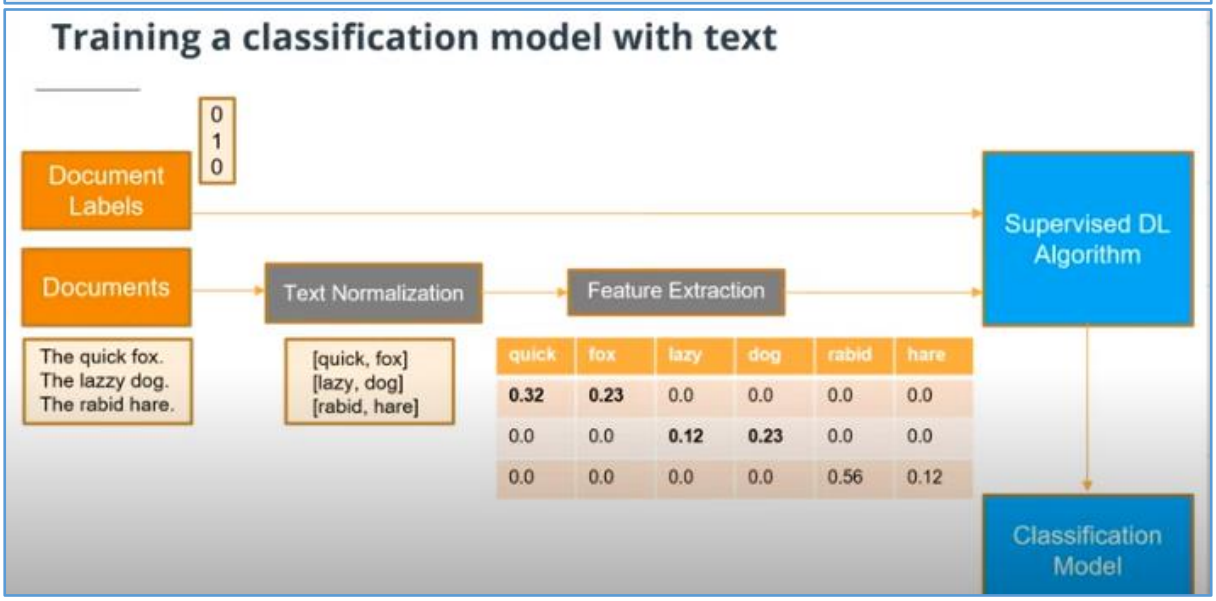
→

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



Chapter 12: Two perspectives of ML

Computer science vs. Statistical perspective

As you can see, data plays a central role in how problems are, modelled 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. Therefore, 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

In the end, having an understanding of the underlying concepts is more important than memorizing the terms used to describe those concepts. However, it is still essential to be familiar with the terminology so that you do not get confused when talking with people from different backgrounds. Over the next couple of pages, we will take, a look at these two different perspectives and get familiar with some of the related terminology.

Chapter 13: 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 will 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 will dive into 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*

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*

Input features and **output features**

A group of input variables is called an **input vector**.

We are learning a program from the input data to make predictions:

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

Chapter 14: Statistical perspective

In statistics, you will 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)
```

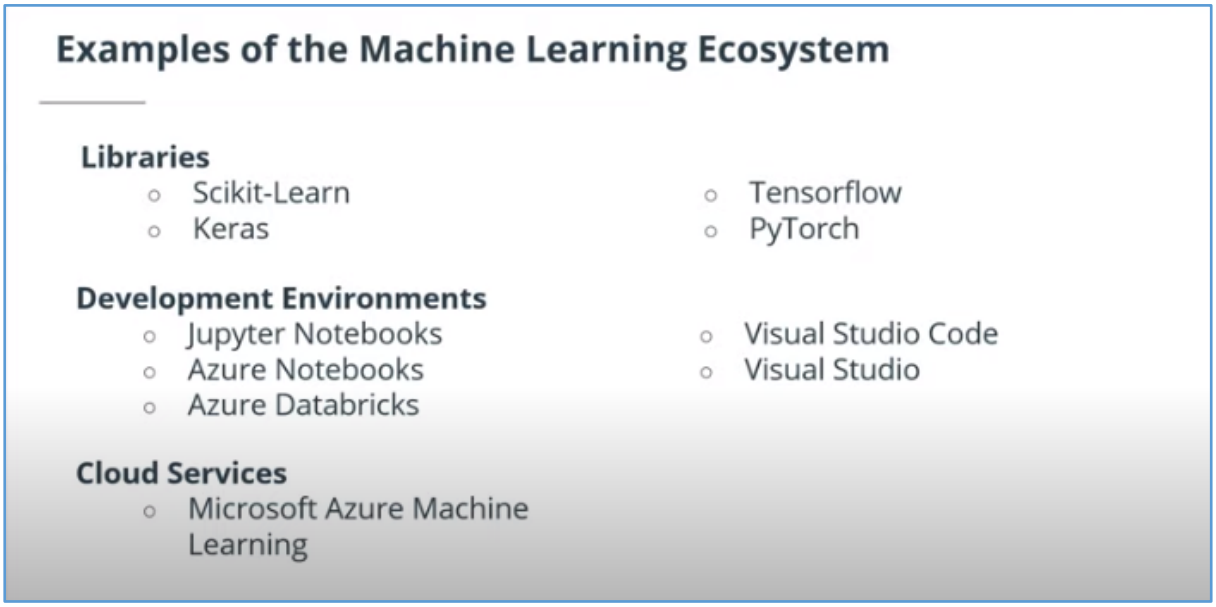
In the case of multiple input variables, X would be an **input vector**, meaning that it would be composed of multiple individual inputs (e.g. `(908721, Guess, Blue, 789, 45.33)`). When this is the case, you will see the individual inputs denoted with a subscript, as in X_1, X_2, X_3, and so on.

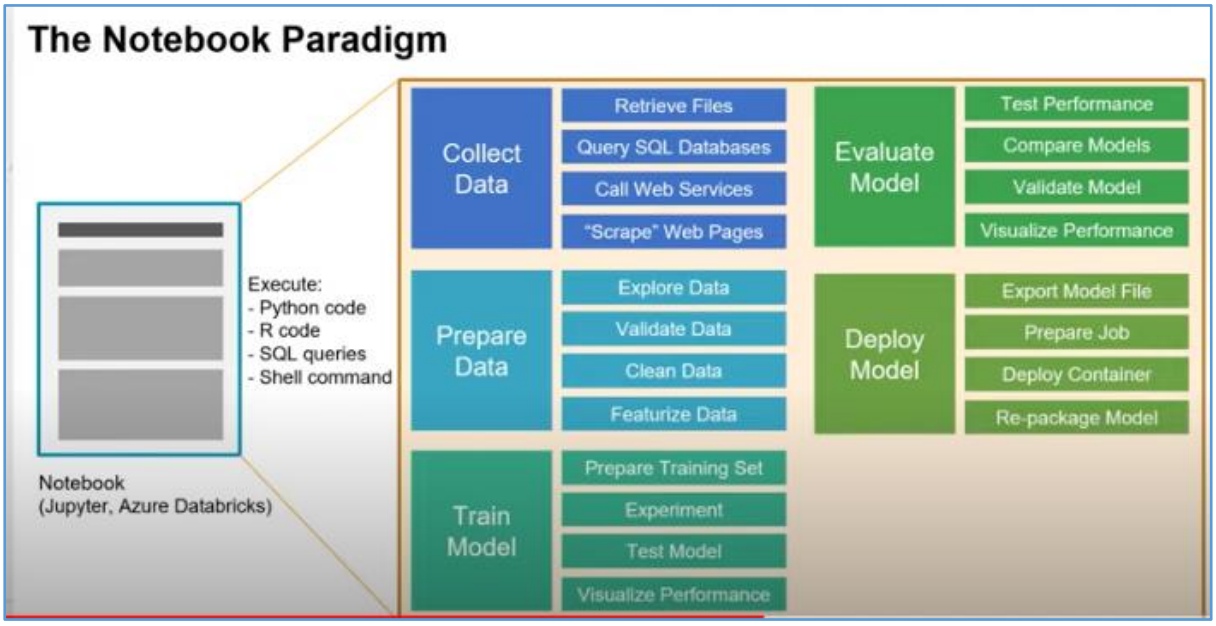
Chapter 15: The Tools for Machine Learning

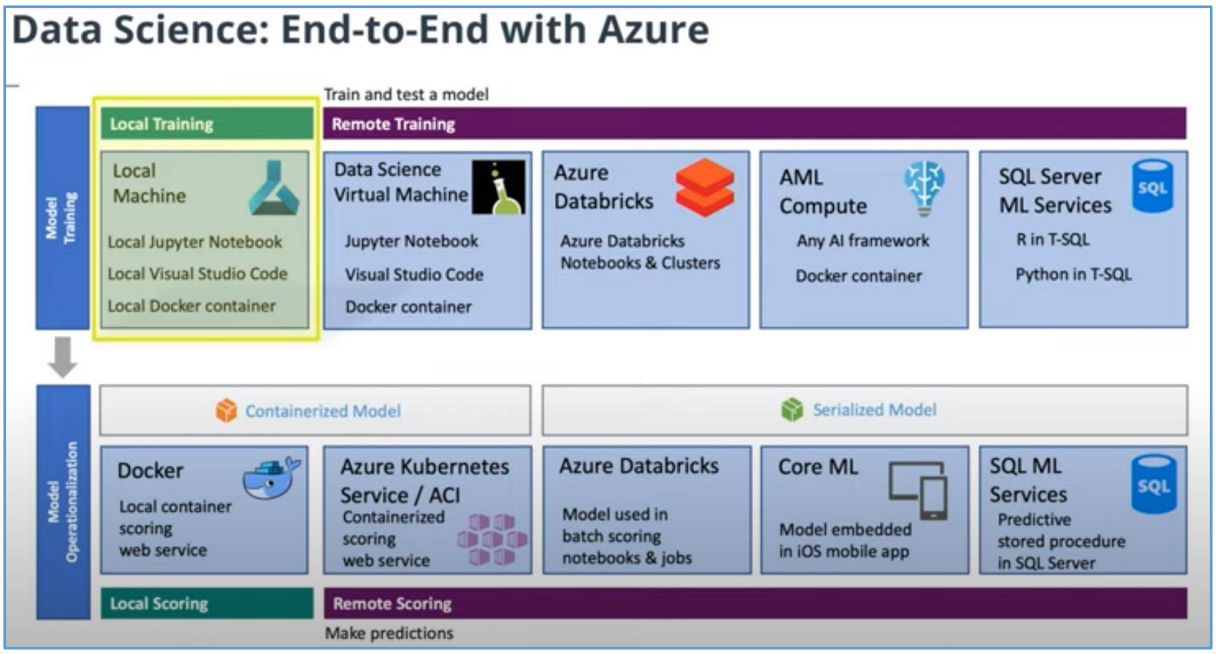
We would require Libraries, Development environment and services to provide support for the entire development process.

Scikit-Learn is a ML library and Tensor Flow, Keras and PyTorch are deep learning libraries.

In case of Jupyter Notebooks, we use python to write codes. Microsoft Azure ML is a Cloud service, which helps in the development process.







Many tools have been, developed to make machine learning more powerful and easier to implement. On this page, we will take, a look at the typical components you might employ in a machine-learning ecosystem. You do not need to understand the details of these tools at this stage, and we do not assume you have 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 are working on a machine-learning project, you likely will not want to write all of the necessary code yourself—instead, you will want to make use of code that has already, been created and refined. That is 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.

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.
For each of these components, there are multiple options, you can choose from. Let us 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. Same can be viewed by others.

End-to-end with Azure

You can analyse and train a small amount of data with your local machine using Jupyter notebook, Visual studio, or other tools. However, with very large amounts of data, or you need a faster processor, it is 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 remotely
For **Model operationalization,** we can test and debug containerized models locally with Docker.
Use **Azure Kubernetes** service or **Azure Container instance** to deploy containerized models in cloud as deployment targets.

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

Chapter 16: Libraries for Machine Learning

Core Framework and Tools

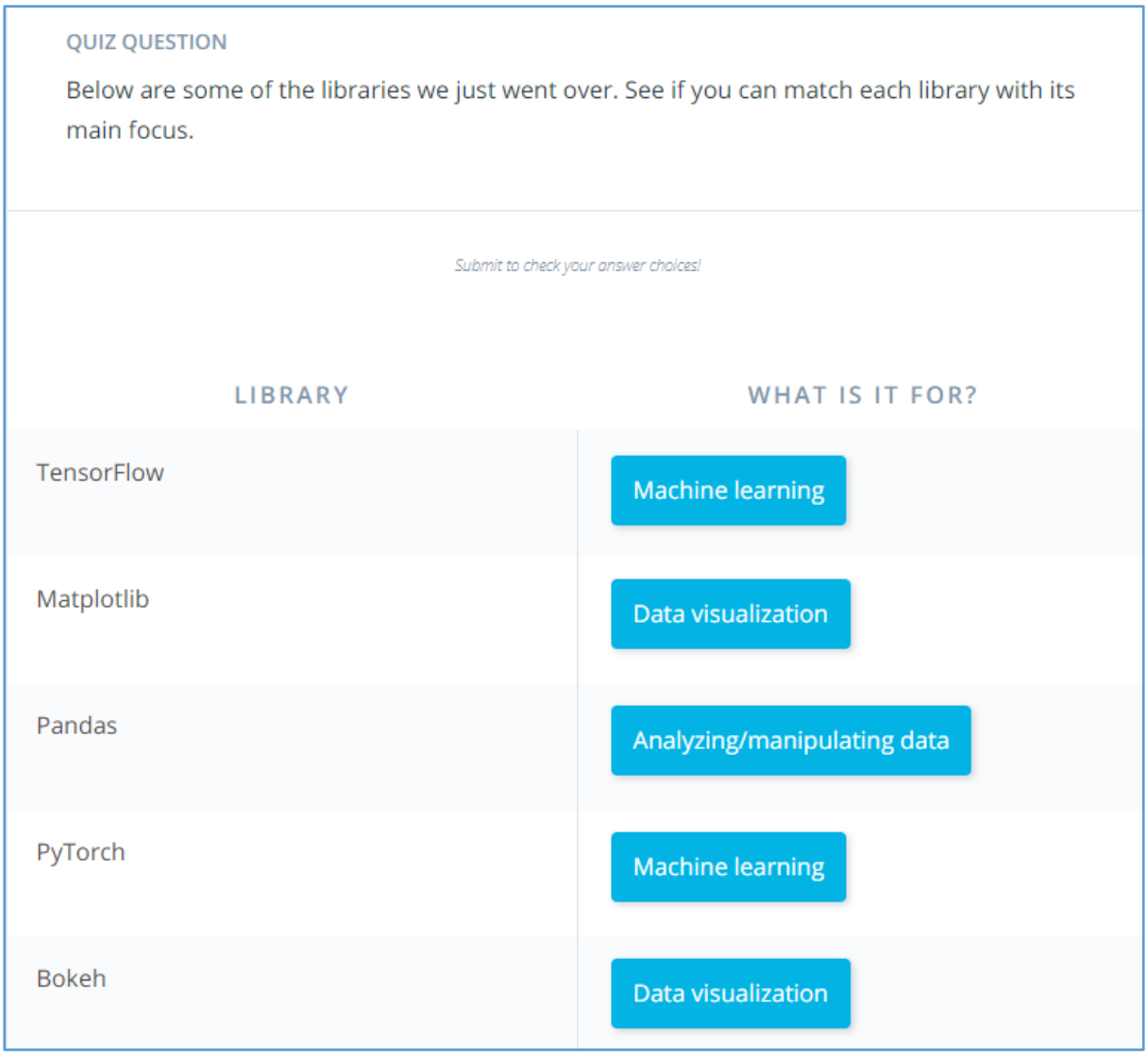
- **Python** is a very popular high-level programming language that is great for data science. Its ease of use and wide support within popular machine learning platforms, coupled with a large catalogue of ML libraries, has made it a leader in this space.
- **Pandas** is an open-source **Python library** 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. **[Numerical optimization library]**
- **Jupyter Notebooks:** Typical environment where we use Python & it ties them together.

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 **[Low-level library & difficult to use]**, 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 will 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.



Chapter 17: 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 will 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:

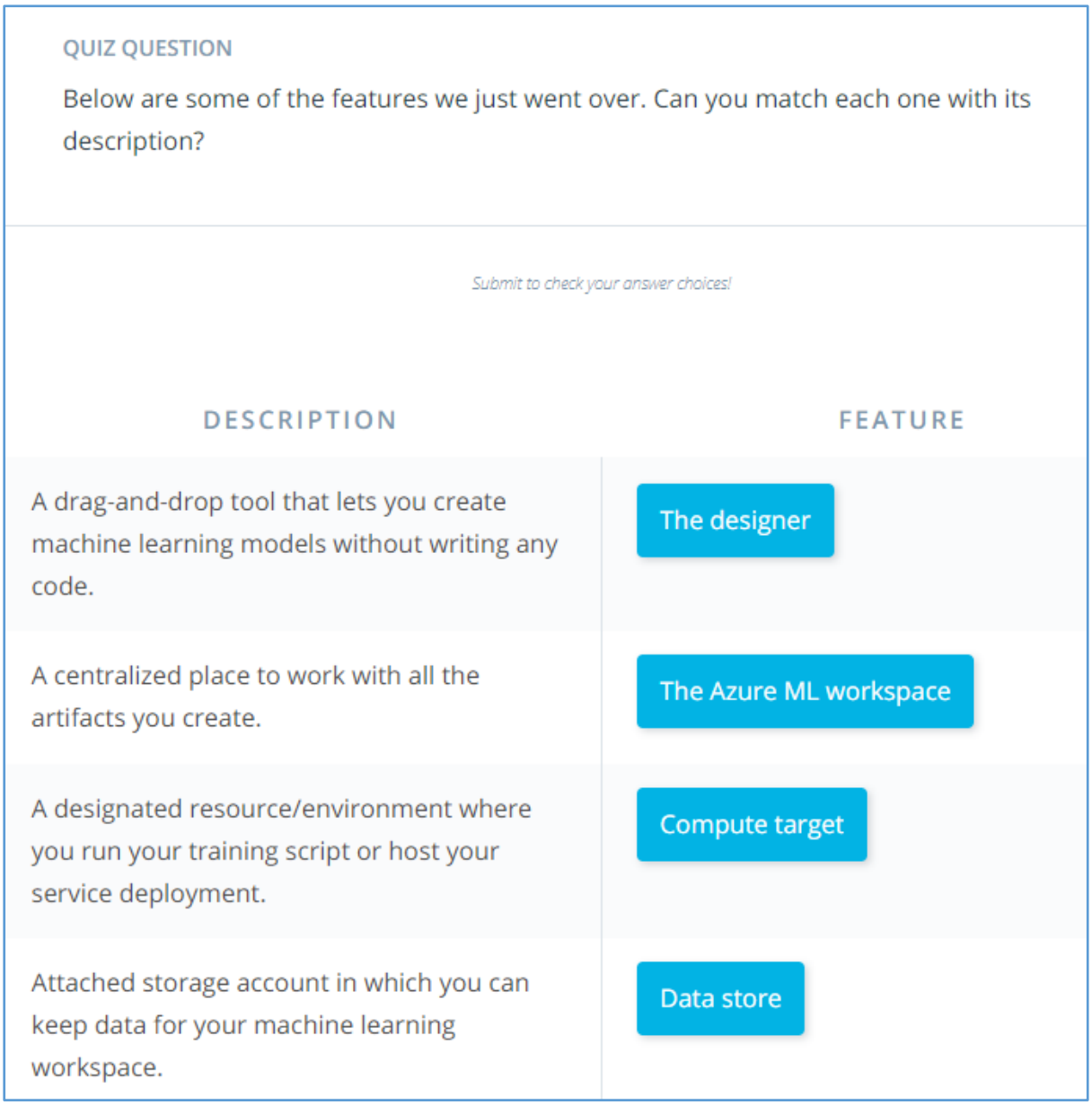
Feature	Description
Compute	Manage compute resources used by machine learning tasks.
Environments	Templates for standardized environments used to create compute resources.
Datastores	Data sources connected to the service environment (e.g. blob stores, file shares, Data Lake stores, and databases).

A Brief Intro to Azure Machine Learning

Below are some of the features of Azure Machine Learning that we just discussed. We will 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.
Experiments	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.



Chapter 18: Model 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:

Model = Algorithm (Data)

On the next page, we will 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. However, 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:

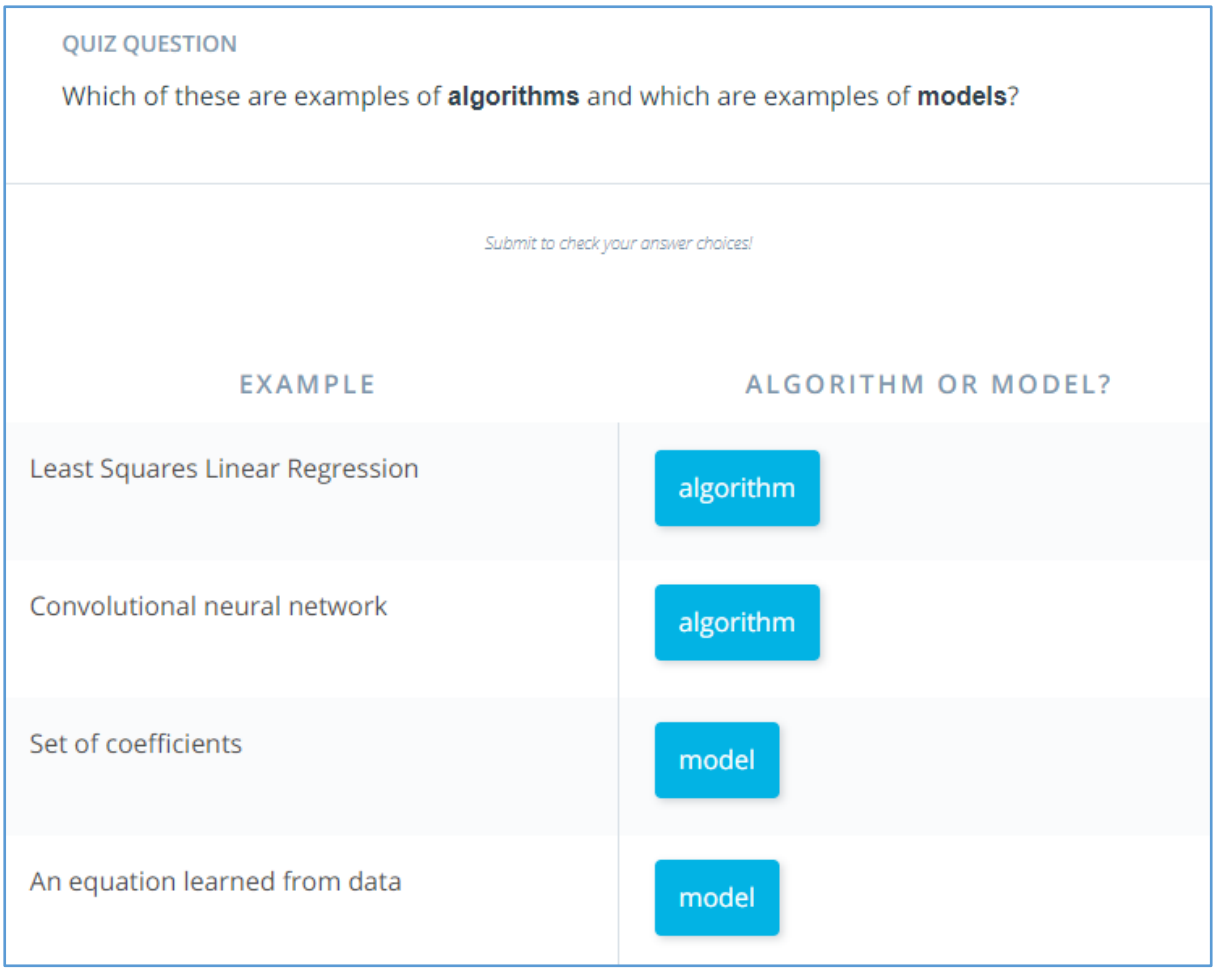
x	y
1	1
2	2
3	3

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 of* W *and* b. The output of the learning process is $W = 1$ and $b = 0$.

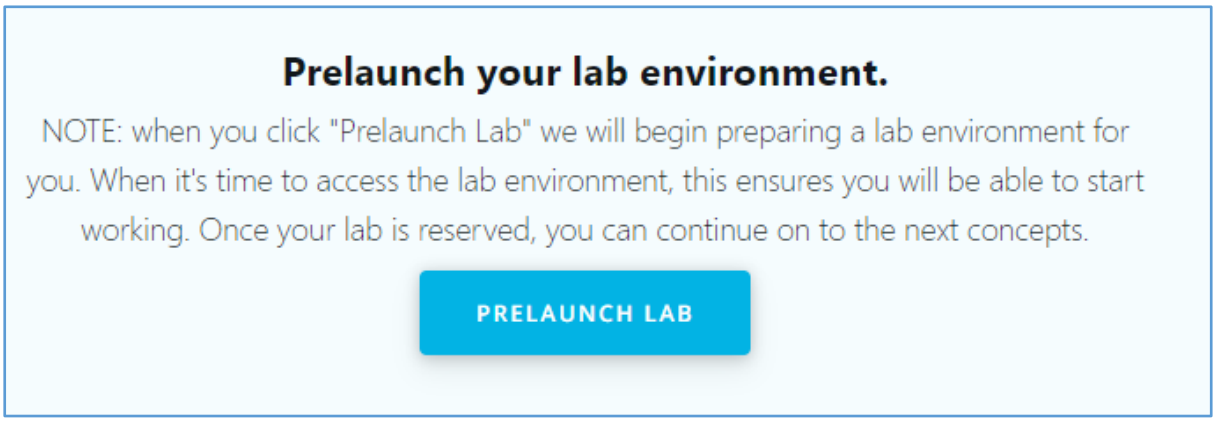
Machine Learning Models

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.



Chapter 19: Prelaunch Lab



Chapter 20: Linear Regression

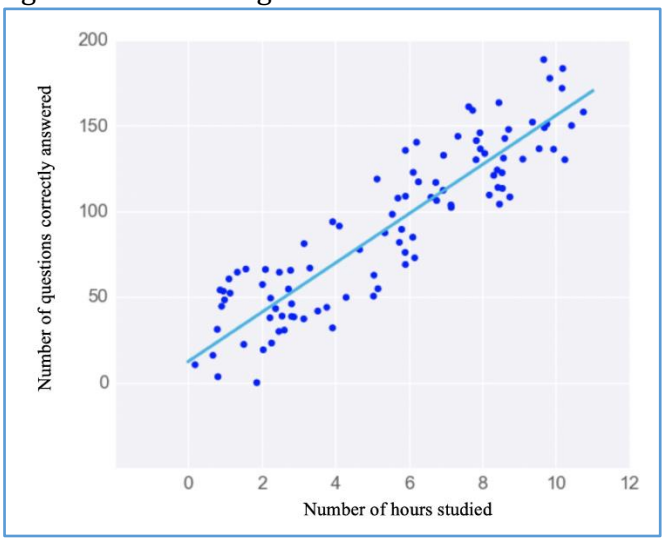
In our first lab, we are going to use Azure Machine Learning Studio to train a model using one of the fundamental machine learning algorithms: *Linear regression*. Before we dive into the lab, let us review what linear regression is and how it can be, used to train a model.

The video below gives a brief review of the main concepts. If you have 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 will 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 us 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**. In addition, 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

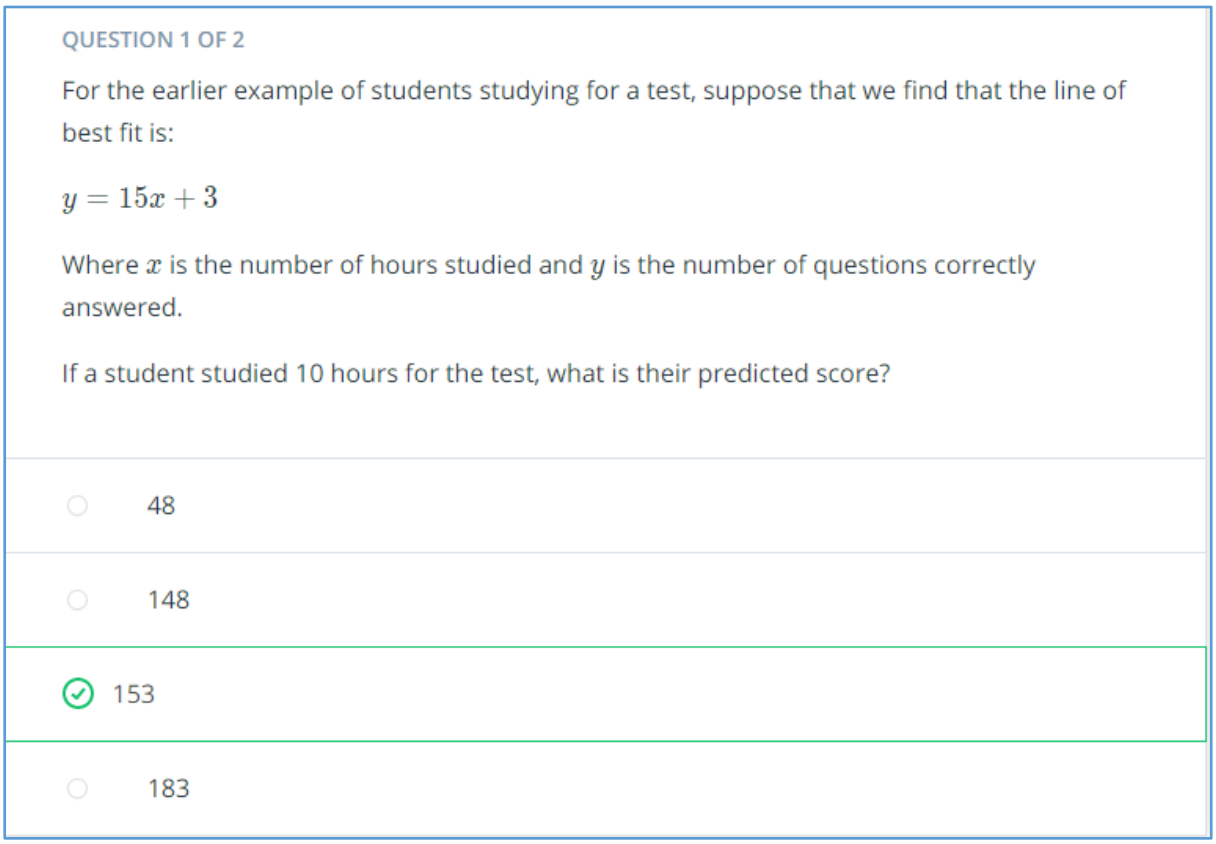
You may recall from fundamental algebra that the general equation for a line looks like this:

$$y = mx+b$$

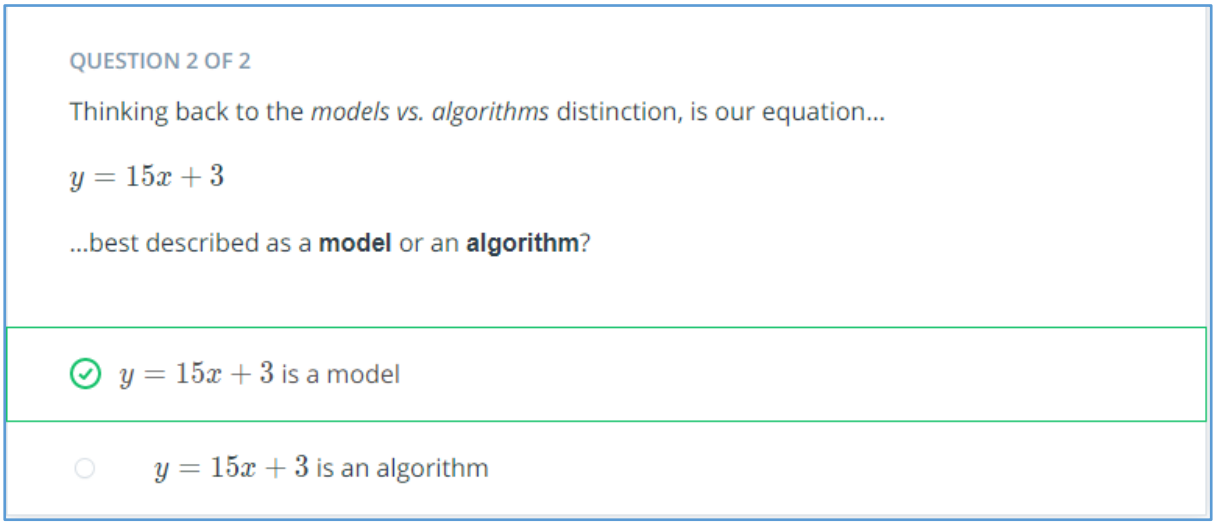
Where m is, called the *slope* of the line, and *b* is the y-intercept. Again, this is the *general* equation. For a specific line, we need to know the values for the slope and y-intercept. For example, the following equations represent three different lines.

$$y = 10x+50$$
$$y = 2x+3$$
$$y = -10x+40$$

Equations like these can be, used to make *predictions*. **Once we know m and b, we can feed in a value for x and the equation will give us the value of y**.



This *specific* line (with m= 15 and *b* = 3) is a *model* that we got from combining the *general* equation y = *mx+b* with the training data.



Linear Regression in Machine Learning

The equation we used above was:

y = *mx+b*

In algebraic terms, we may refer to m as the **coefficient** of x or simply the **slope** of the line, and we may call 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:

$$y = B_0 + B_1 * x$$

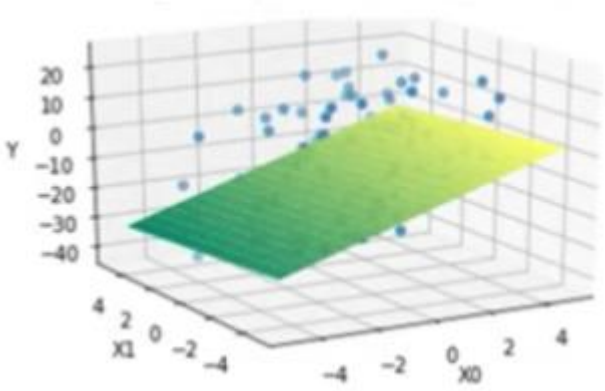
The letters are different and the order has been, changed, but it is exactly the same equation. Thus, we can see that what we know from algebra as the basic equation for a line is also, in machine learning, the equation used for **simple linear regression**.

Multiple Linear Regression

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

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

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:



However, do not let any of this intimidate you: The core idea is still that we are modelling 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

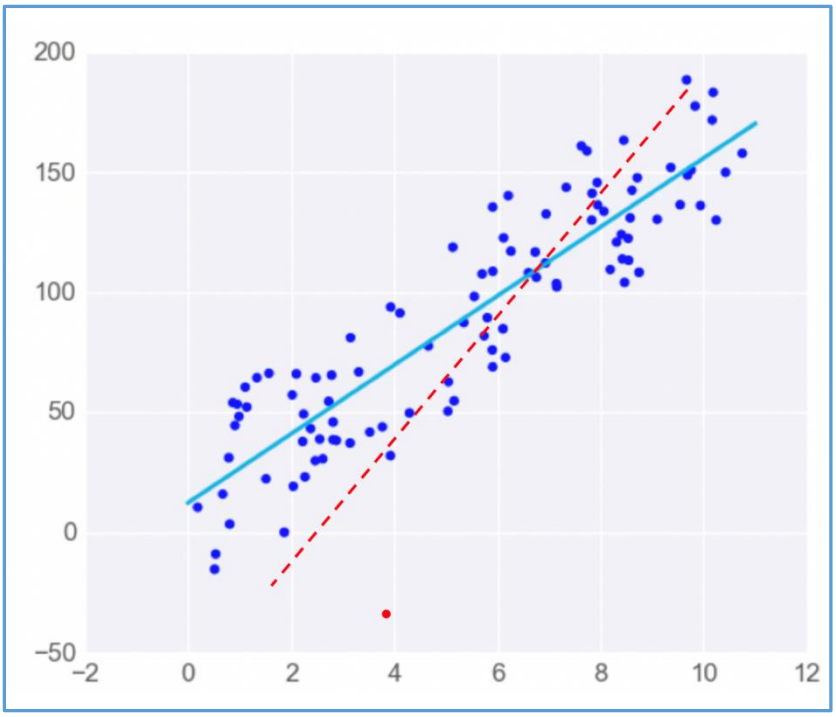
Notice from our example of test scores earlier that the line we came up with did not *perfectly* fit the data. In fact, *most* of the data points were not on the line! When we predict that a student who studies for 10 hours will get a score of 153, we do not expect their score to be exactly 153. Put 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 on 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 have 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, modelled 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 is 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 is 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 have 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 us look at the formulas used to calculate the coefficients. We are 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:

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

p is the predicted value and y is the actual value

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.

Lesson 21: Linear Regression – Check your understanding

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

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

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

QUESTION 4 OF 4

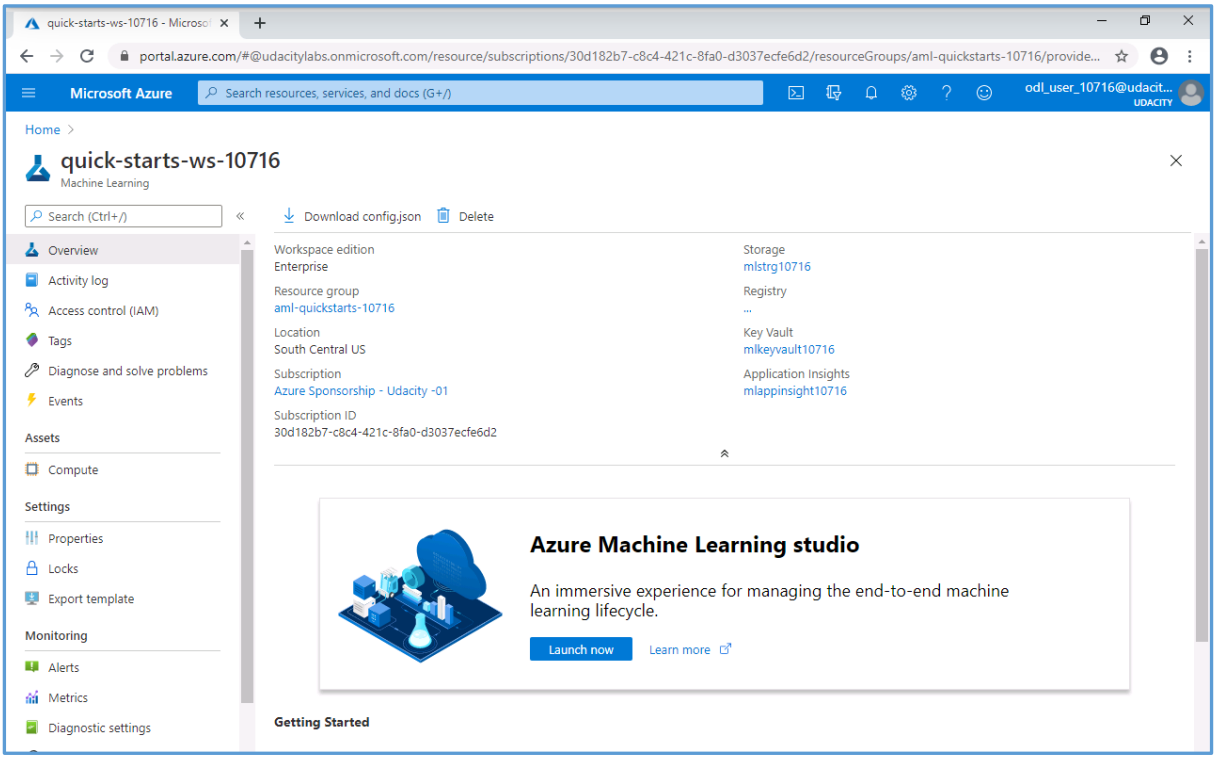
Which one of the following is not necessary when preparing data for a linear regression model?

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

While preparing data we look into Linear Relationship, Collinearity, Rescale data, Remove outliers and Gaussian distribution of residuals.

Minimization of errors is done to find the best model.

Chapter 23: Lab- Train a Linear Regression Model



Lab Overview

Azure Machine Learning designer (preview) gives you a cloud-based interactive, visual workspace that you can use to easily and quickly prep data train and deploy machine-learning models. It supports Azure Machine Learning compute, GPU or CPU. Machine learning designer also supports publishing models as web services on Azure Kubernetes Service that can easily, be consumed by other applications.

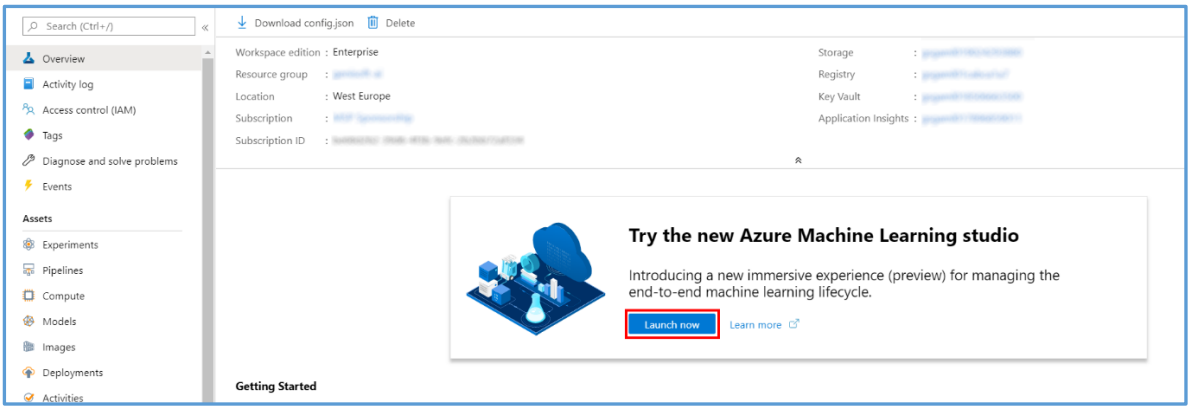
In this lab, we will be using a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets. The data is, enriched with holiday and weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a regression model to predict NYC taxi fares. To train the model, we will create Azure Machine Learning Compute resource. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

Exercise 1: Register Dataset with Azure Machine Learning studio

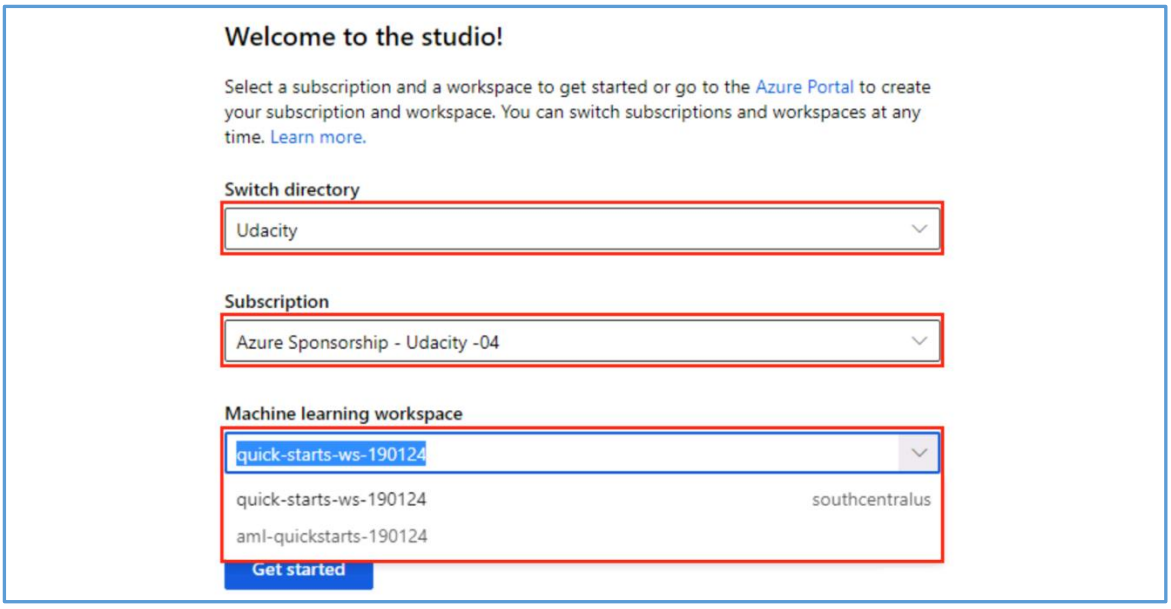
Task 1: Upload Dataset

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

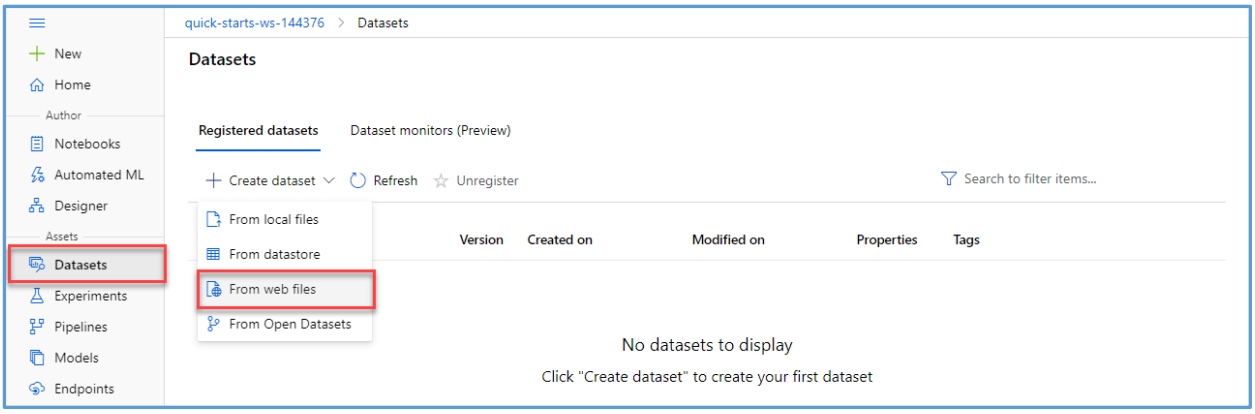


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it does not matter which) and then click **Get started**.

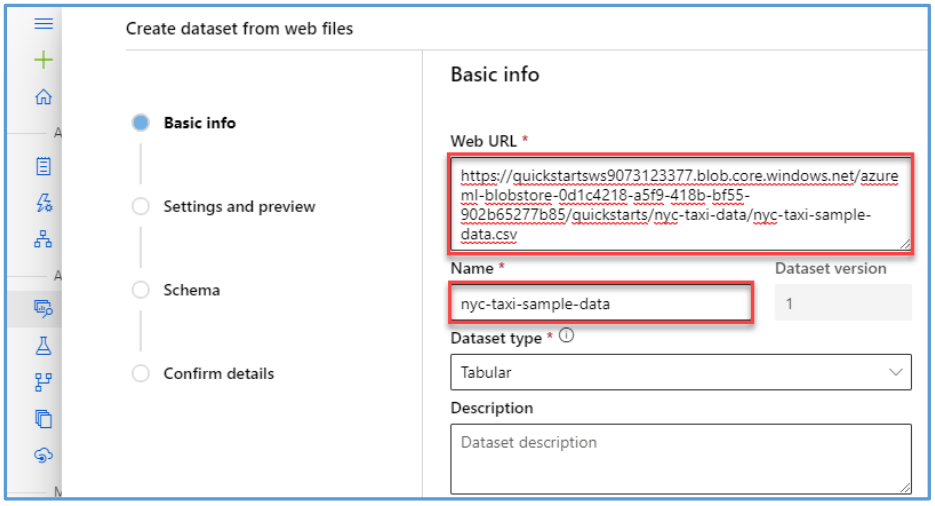
4. From the studio, select **Datasets, + Create dataset, from web files**. This will open the Create dataset from web files dialog on the right.



5. In the Web URL field provide the following URL for the training data file:

```
https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data.csv
```

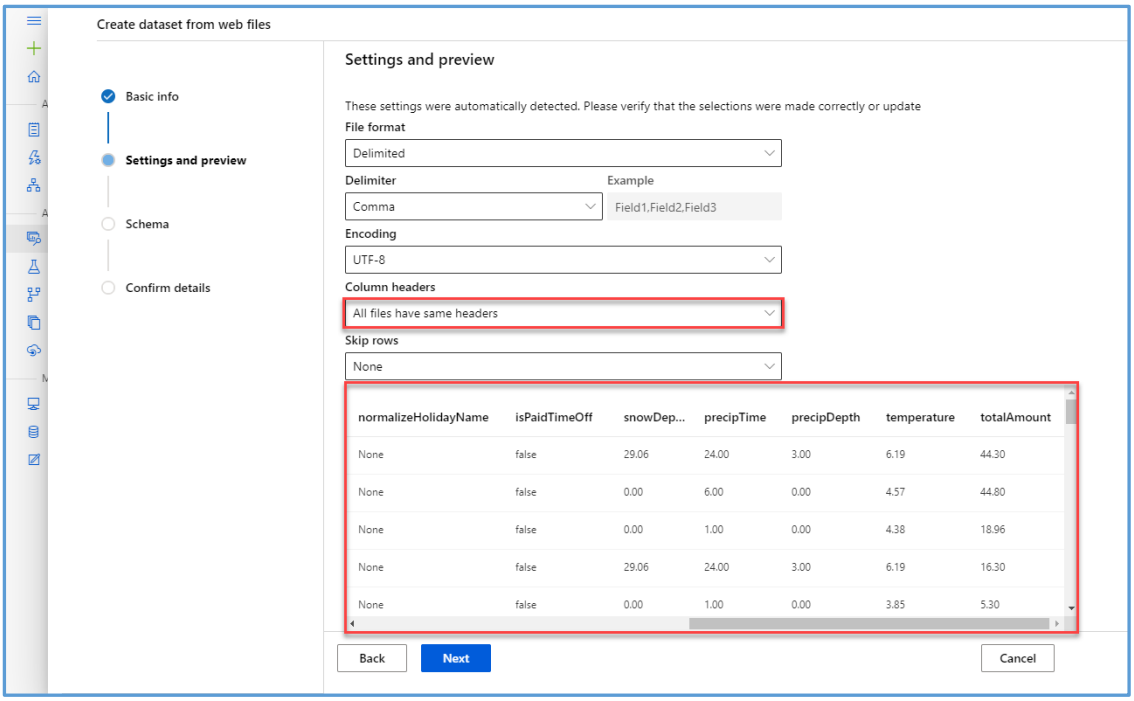
6. Provide `nyc-taxi-sample-data` as the Name, leave the remaining values at their defaults and select **Next**.



Task 2: Preview Dataset

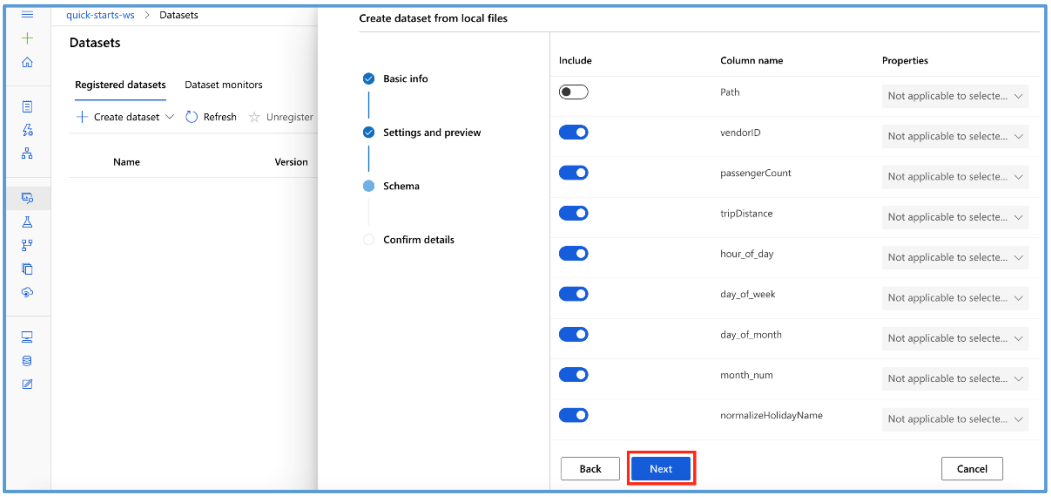
1. On the Settings and preview panel, set the column headers drop down to `All files have same headers`.

2. Scroll the data preview to right to observe the target column `totalAmount`. After you are done reviewing the data, select **Next**



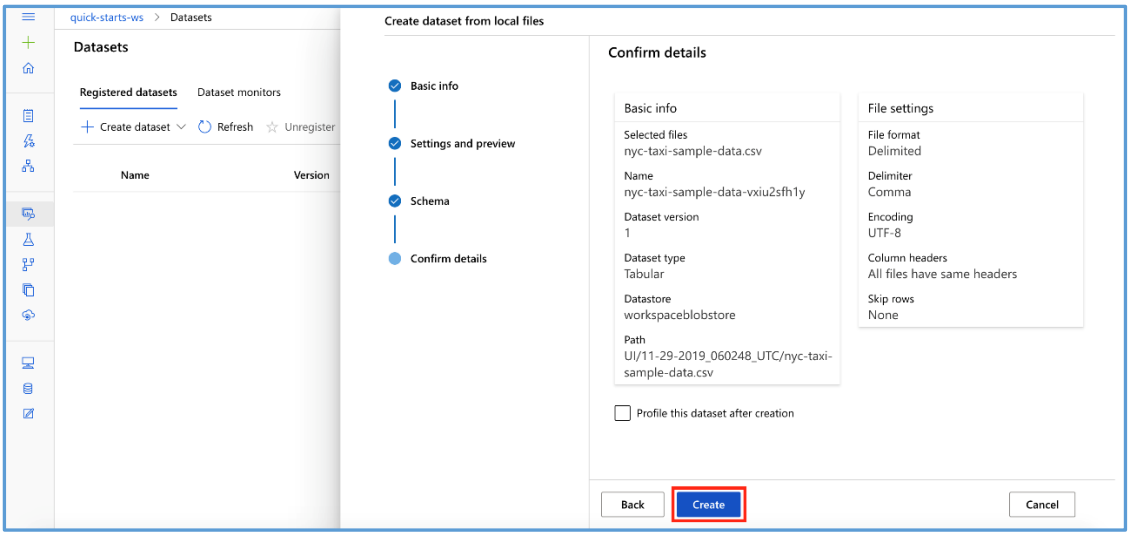
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Leave the default selections and select **Next**



Task 4: Create Dataset

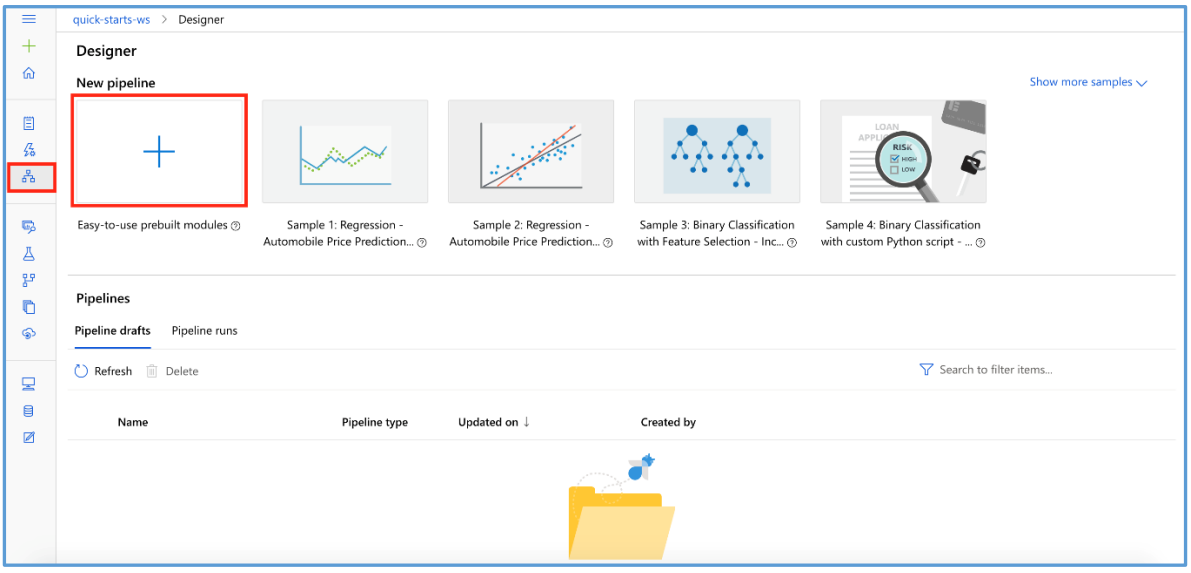
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

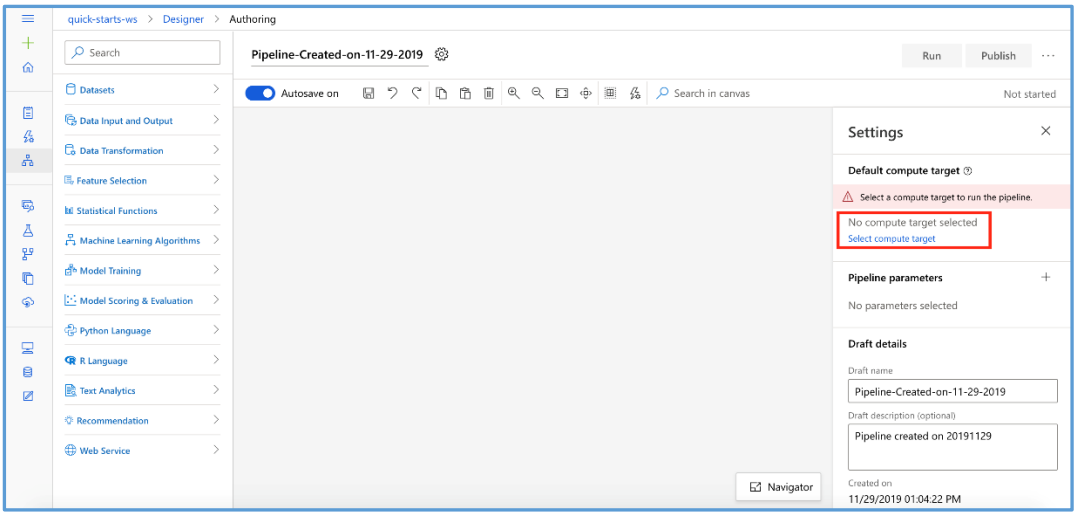
Task 1: Open Pipeline Authoring Editor

1. From the studio, select **Designer, +**. This will open a visual pipeline authoring editor.



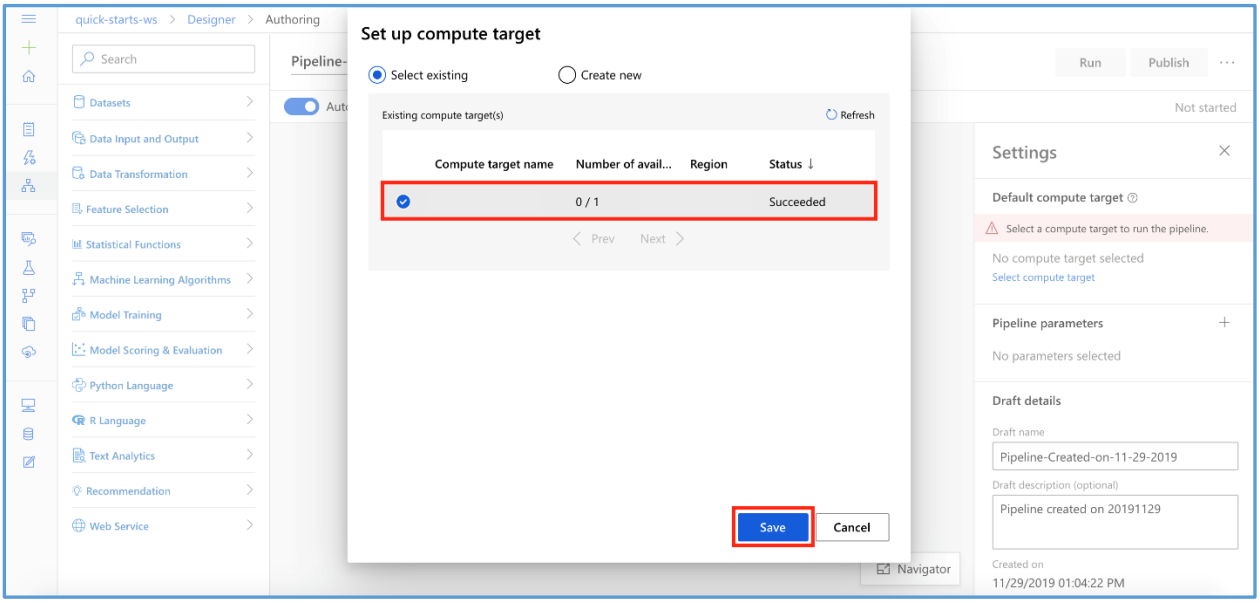
Task 2: Setup Compute Target

1. In the settings panel on the right, select **Select compute target**.



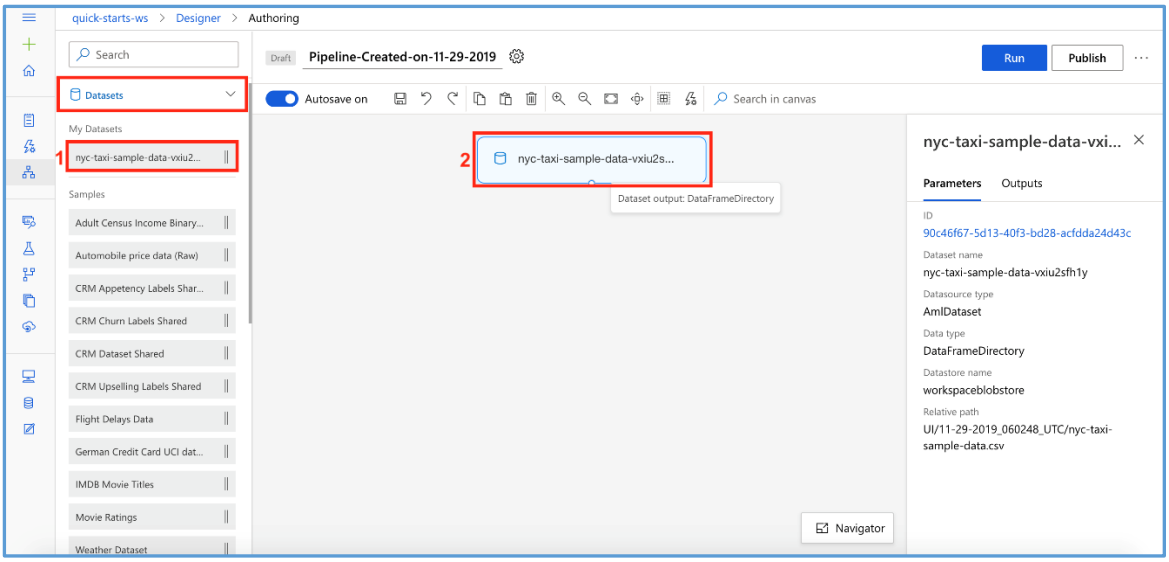
2. In the Set up compute target editor, select the available compute, and then select **Save**.

Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.



Task 3: Add Dataset

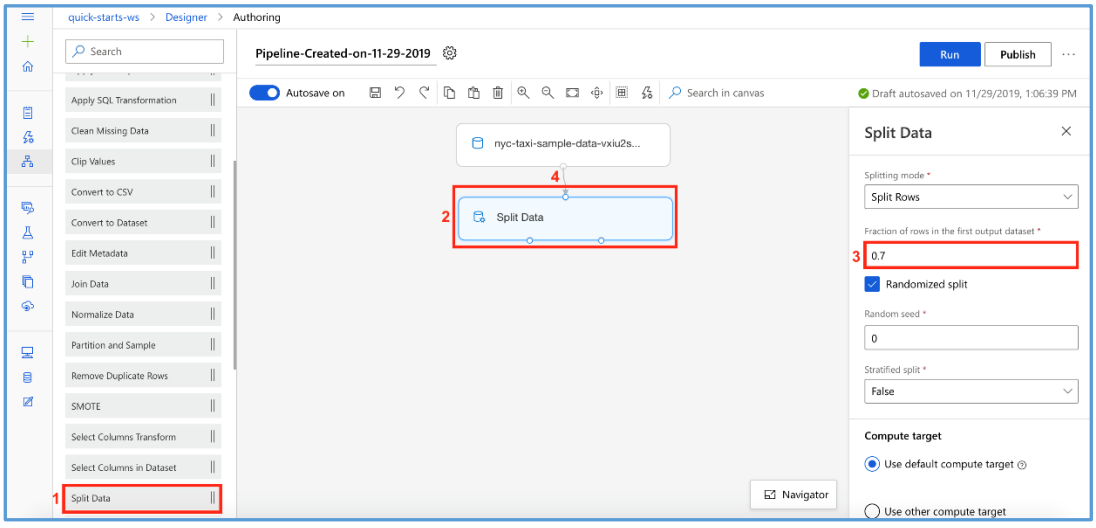
1. Select **Datasets** section in the left navigation. Next, select **My Datasets, nyc-taxi-sample-data** and drag and drop the selected dataset on to the canvas.



Task 4: Split Dataset

1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Split Data** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Fraction of rows in the first output dataset: **0.7**
    4. Connect the Dataset to the Split Data module



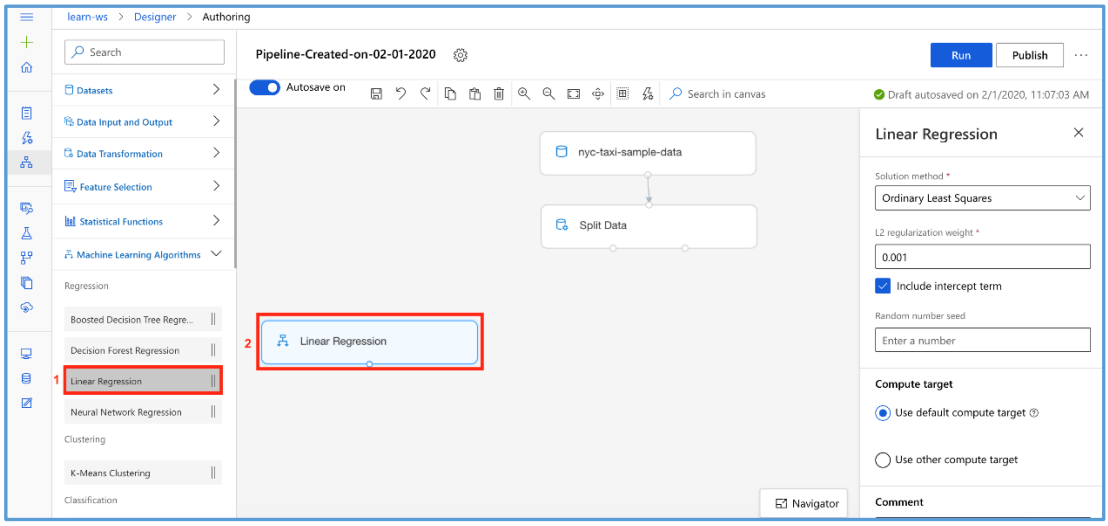
Note that you can submit the pipeline at any point to peek at the outputs and activities. Running pipeline also generates metadata that is available for downstream activities such selecting column names from a list in selection dialogs.

Task 5: Initialize Regression Model

1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:

    1. Select the **Linear Regression** prebuilt module

    2. Drag and drop the selected module on to the canvas



Task 6: Setup Train Model Module

1. Select **Model Training** section in the left navigation. Follow the steps outlined below:

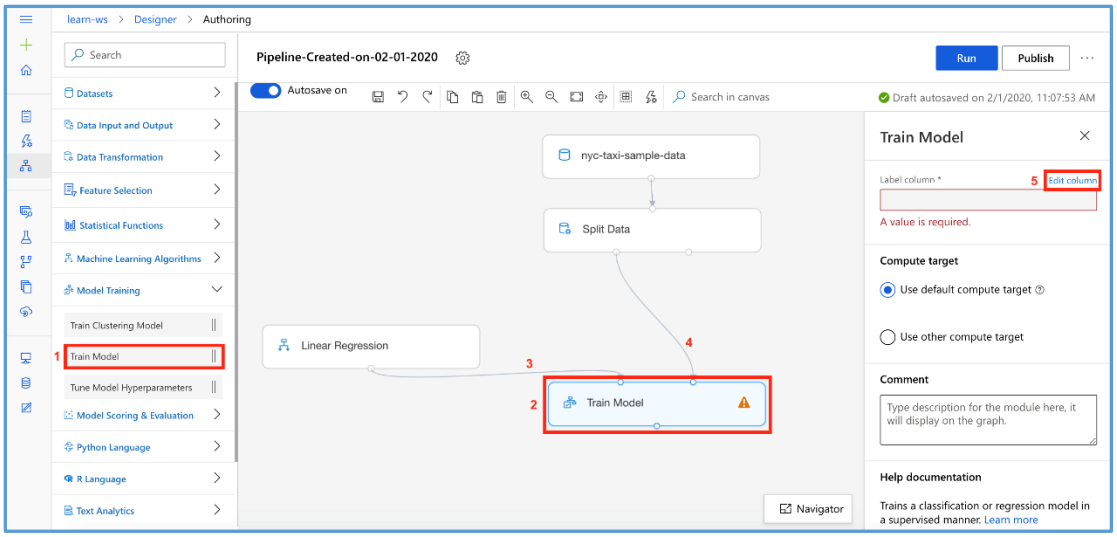
    1. Select the **Train Model** prebuilt module

    2. Drag and drop the selected module on to the canvas

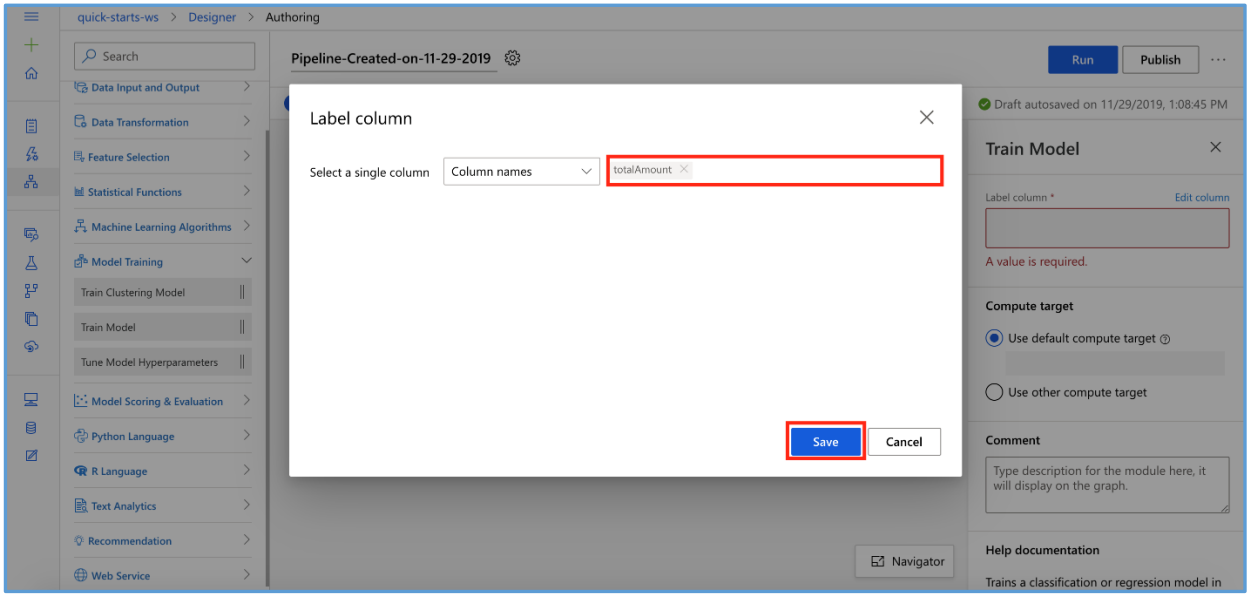
    3. Connect the `Linear Regression` module to the first input of the `Train Model` module

    4. Connect the first output of the `Split Data` module to the second input of the `Train Model` module

    5. Select the **Edit column** link to open the `Label column` editor



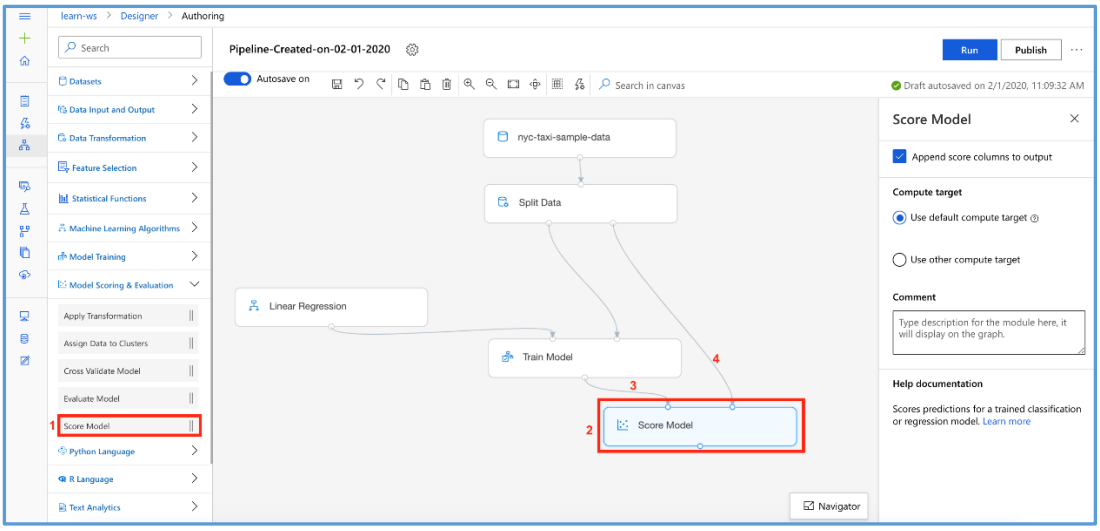
2. The Label column editor allows you to specify your Label or Target column. Type in the label column name **totalAmount** and then select **Save**.



Task 7: Setup Score Model Module

1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the Train Model module to the first input of the Score Model module
    4. Connect the second output of the Split Data module to the second input of the Score Model module



Note that Split Data *module will feed data for both model training and model scoring. The first output (0.7 fraction) will connect with the* Train Model *module and the second output (0.3 fraction) will connect with the* Score Model *module.*

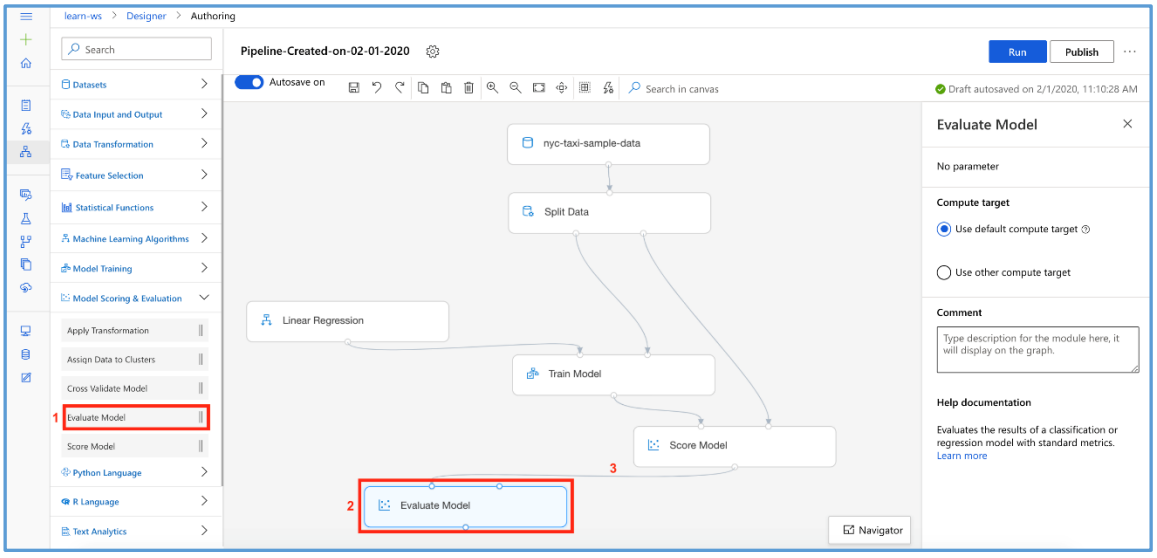
Task 8: Setup Evaluate Model Module

1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Evaluate Model** prebuilt module

    2. Drag and drop the selected module on to the canvas

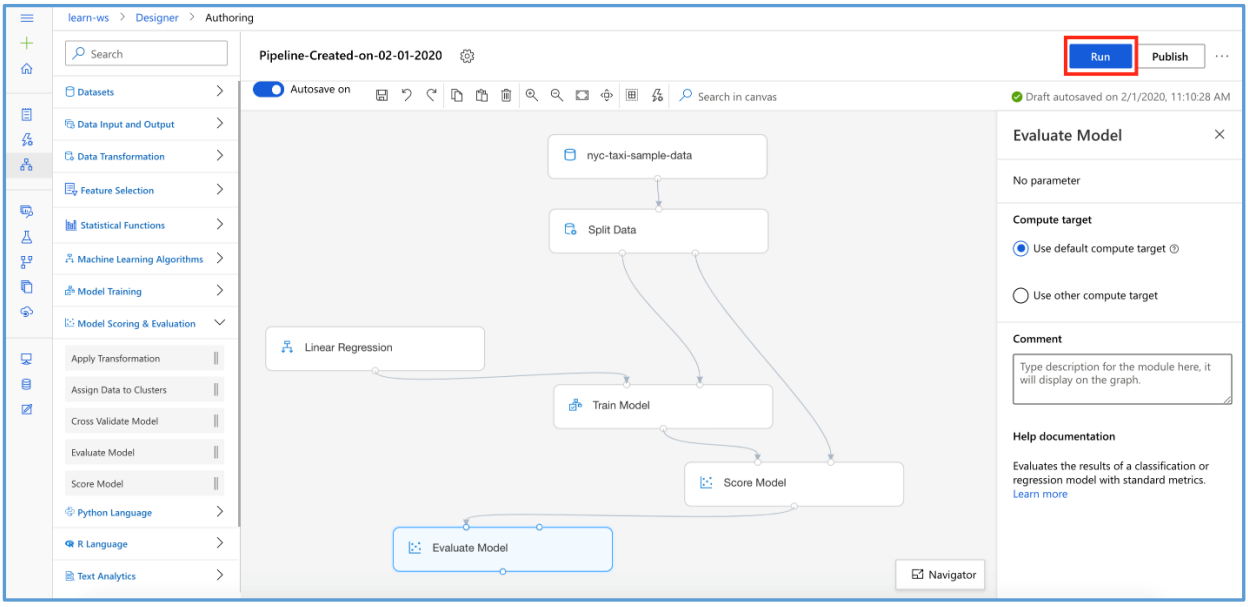
    3. Connect the Score Model module to the first input of the Evaluate Model module



Exercise 3: Submit Training Pipeline

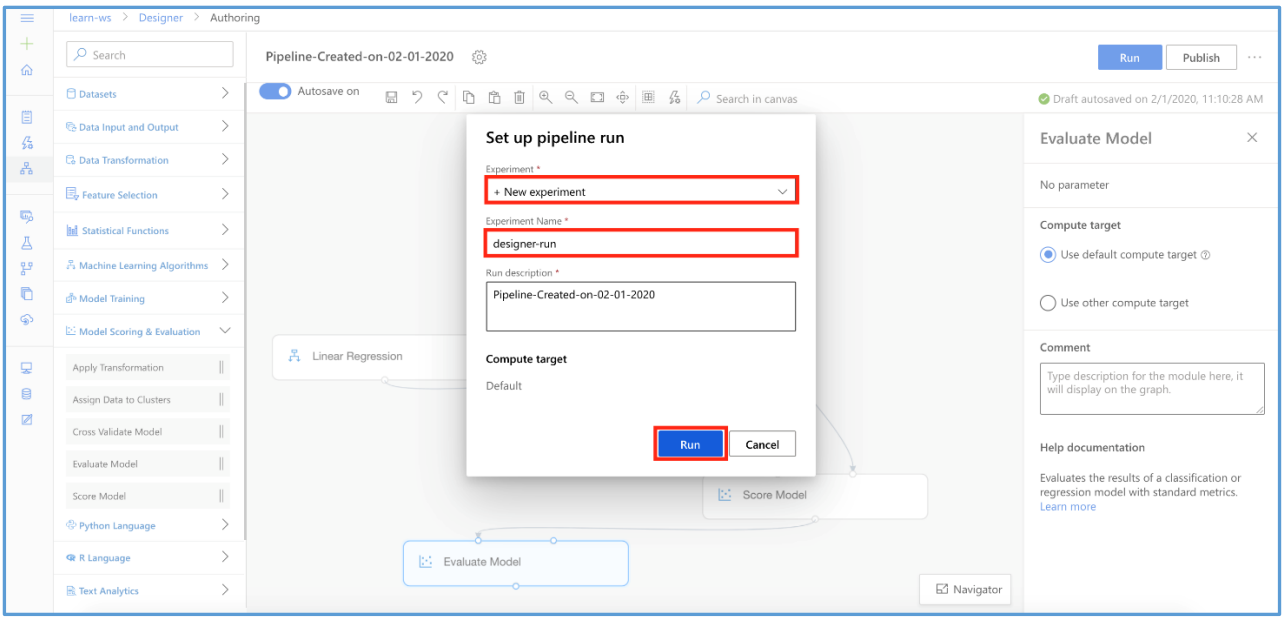
Task 1: Create Experiment and Submit Pipeline

1. Select **Submit** to open the Setup pipeline run editor.



Please note that the button name in the UI is, changed from **Run** to **Submit**.

2. In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **designer-run**, and then select **Submit**.



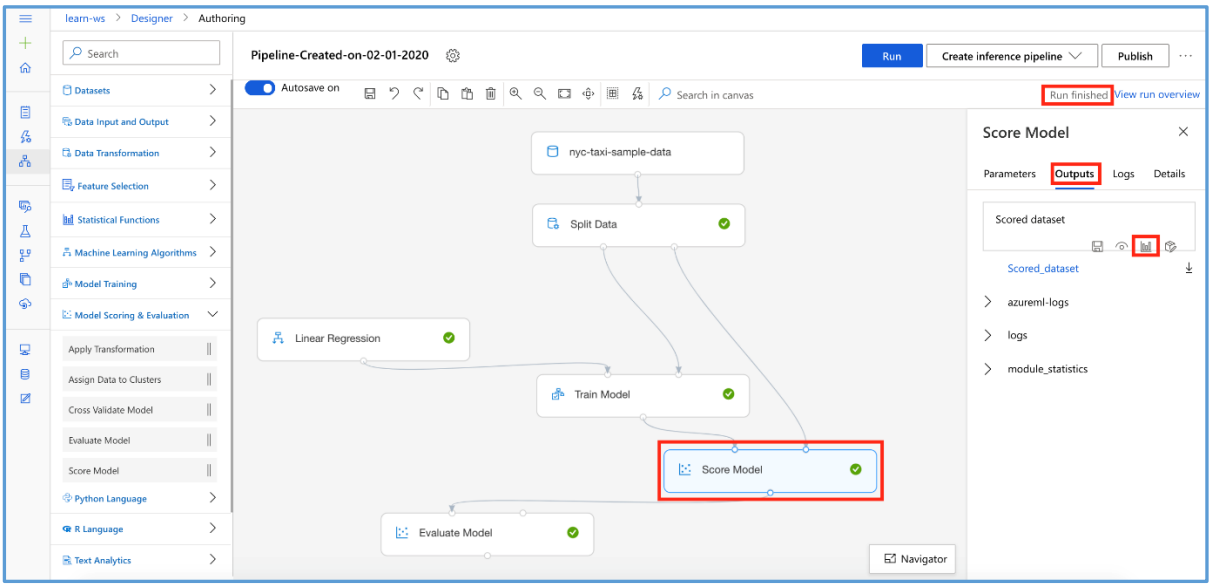
3. Wait for pipeline run to complete. It will take around **8 minutes** to complete the run.

4. While you wait for the model training to complete, you can learn more about the training algorithm used in this lab by selecting Linear Regression module.

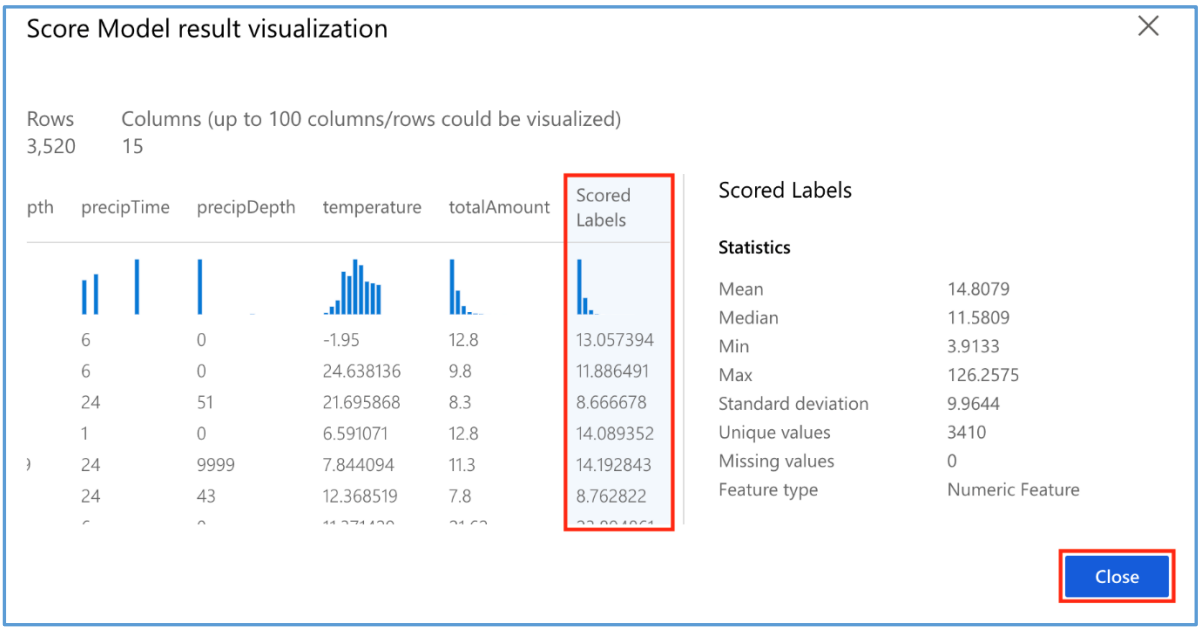
Exercise 4: Visualize Training Results

Task 1: Visualize the Model Predictions

1. Select **Score Model, Outputs, Visualize** to open the Score Model result visualization dialog.

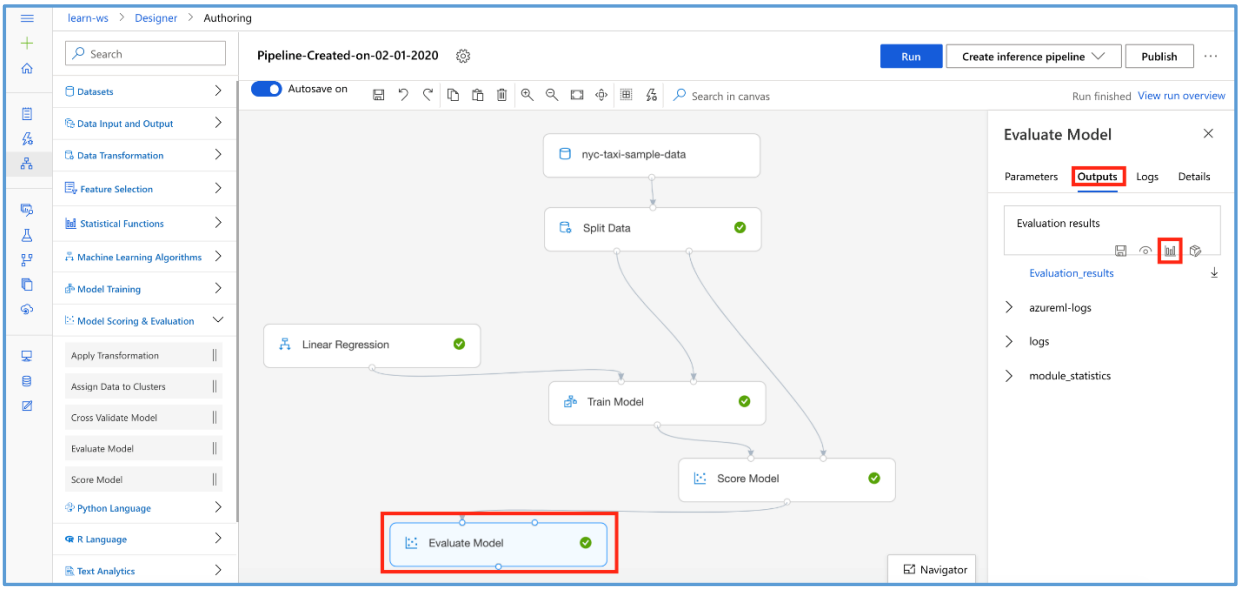


2. Observe the predicted values under the column **Scored Labels**. You can compare the predicted values (Scored Labels) with actual values (totalAmount).

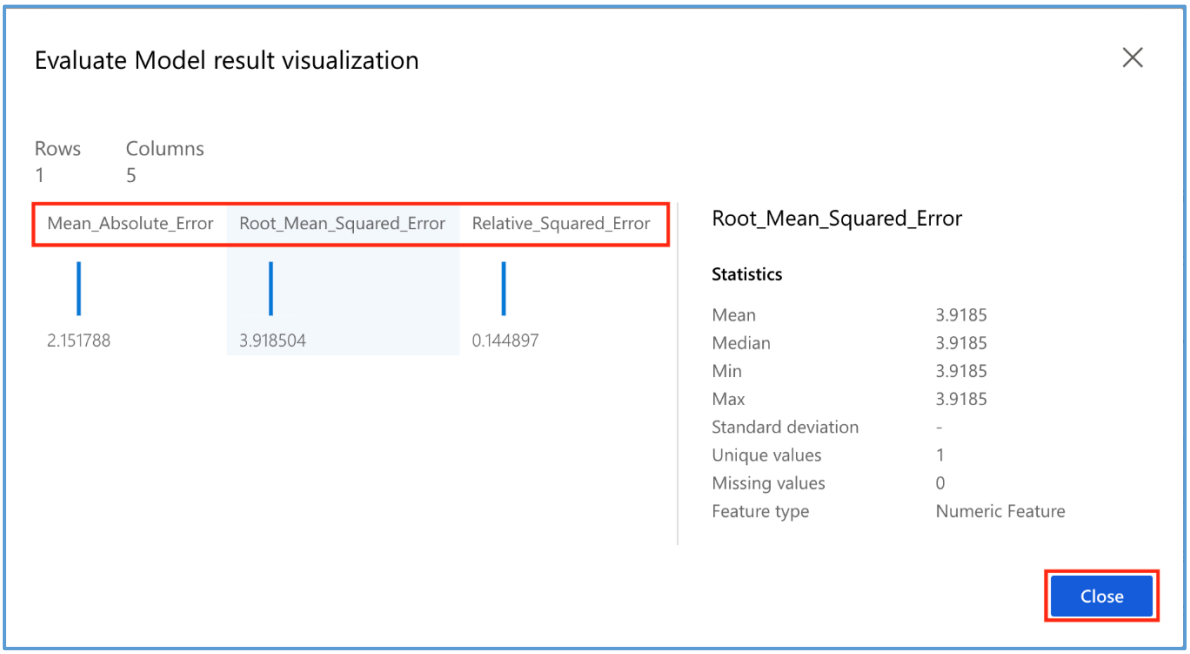


Task 2: Visualize the Evaluation Results

1. Select **Evaluate Model, Outputs, Visualize** to open the Evaluate Model result visualization dialog.



2. Evaluate the model performance by reviewing the various evaluation metrics, such as **Mean Absolute Error**, **Root Mean Squared Error**, etc.

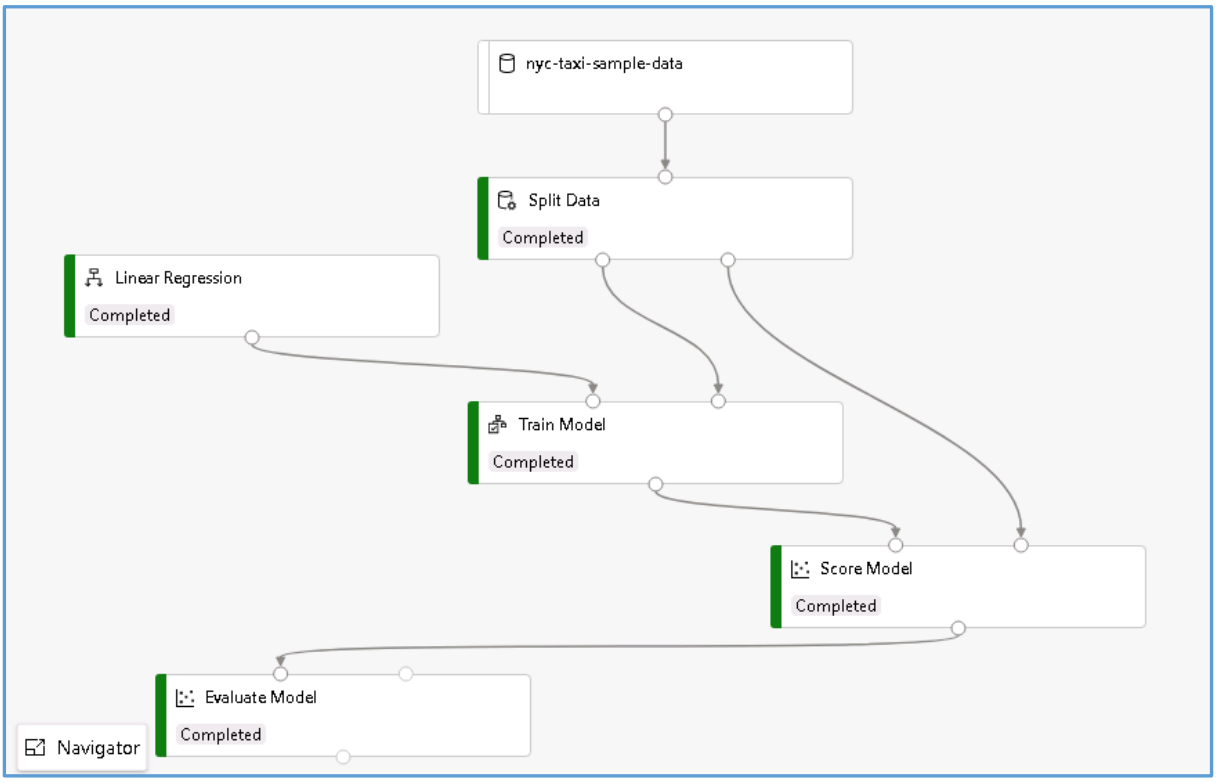


Next Steps

Congratulations! You have trained and evaluated your first machine learning model. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

Chapter 24: Walkthrough - Train a Linear Regression Model

- Compute resource scales automatically as per the need of the experiment.
- Data Transformation >> Split Data (0.7 for Training & 0.3 for Soring)
- ML Algorithm >> Linear regression
- Model Training >> Train Model
- Model Scoring & Evaluation >> Score Model (We get the predicted value w.r.to the actual value)
- Model Scoring & Evaluation >> Evaluate Model (We get the Cost function RMSE values)



Chapter 25: 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) that describes the mapping between data input variables (X) and an output variable (Y).

$Y = f(X)$

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

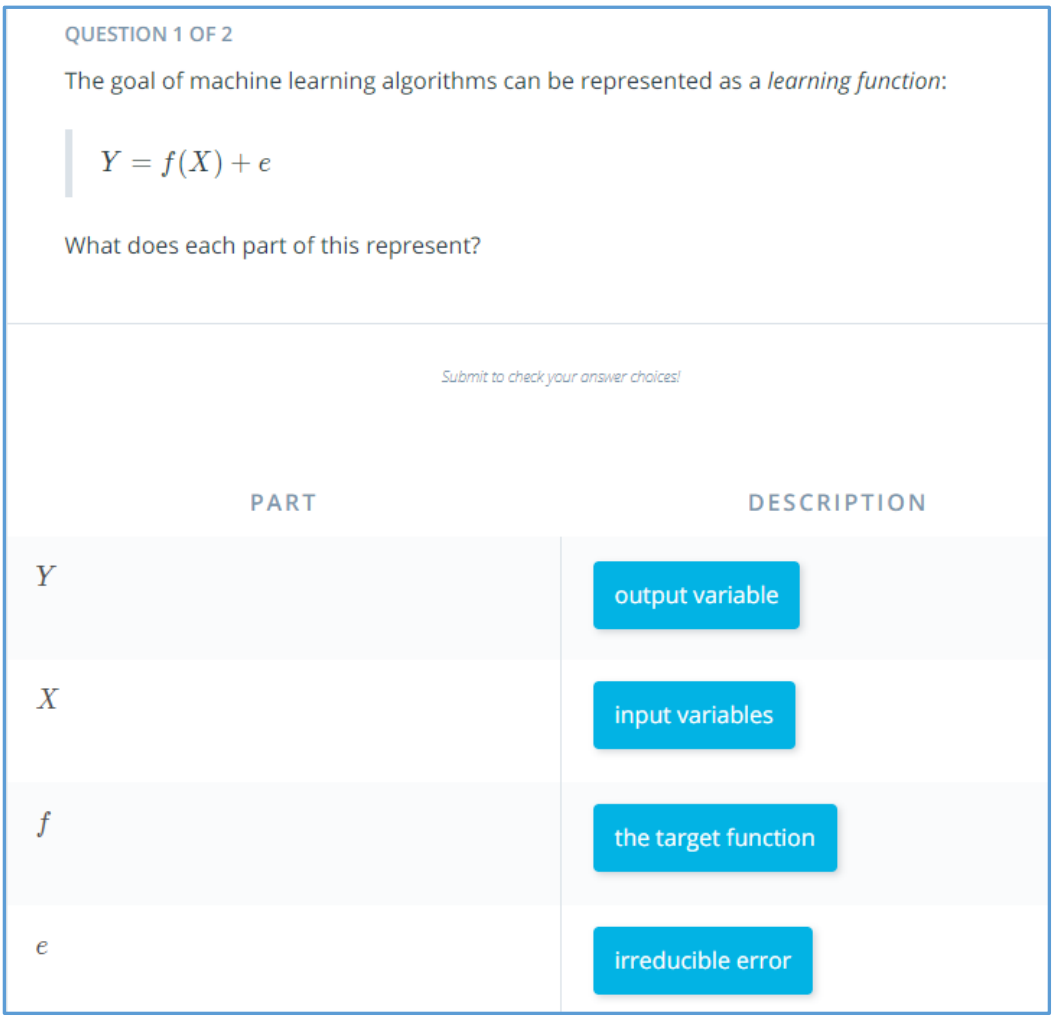
Since the process extrapolates from a limited set of values, there will always be an error e which is, independent of the input data (X) such that:

$Y = f(X) + e$

The variable e is, called **irreducible error** because no matter how good we get at estimating the target function (f), we cannot reduce this error. In addition, e is independent of the input variable X.

Note that the irreducible error we are discussing here is different from the model error we talked about earlier in the lesson. Irreducible error is, caused by data collection process. Such as when we do not have enough data or do not 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 it can be, reduced during the model learning process.



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.

Chapter 26: 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:

$$B_0 + B_1 * X_1 + B_2 * X_2 = 0$$

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.

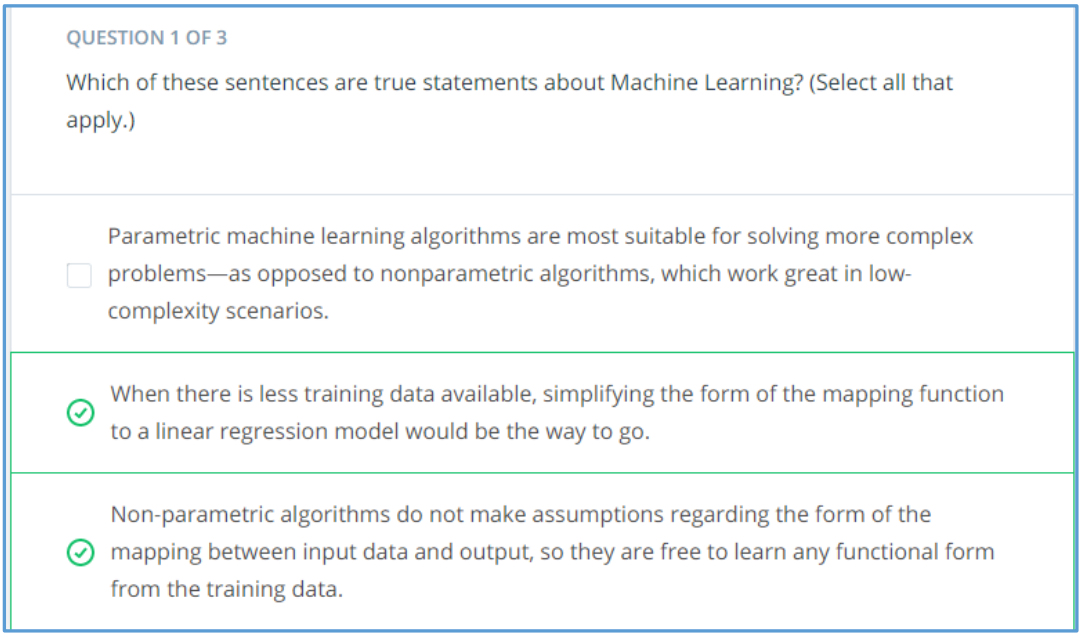
A simple example is the K-nearest neighbours (KNN) algorithm, which we will discuss in more detail later in the course. KNN does not make any assumptions about the functional form, but instead uses the pattern that points have similar output when they are close.

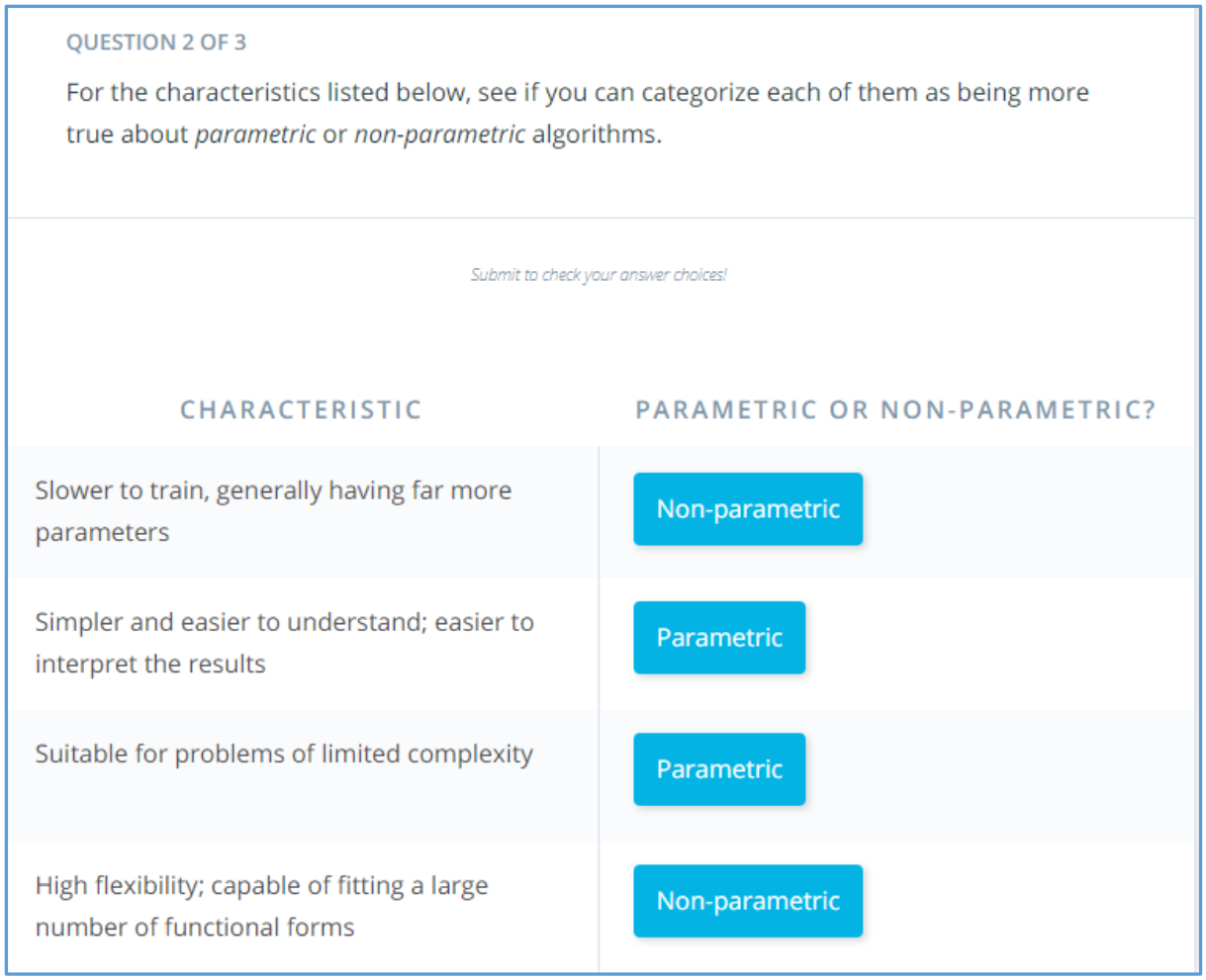
Benefits:

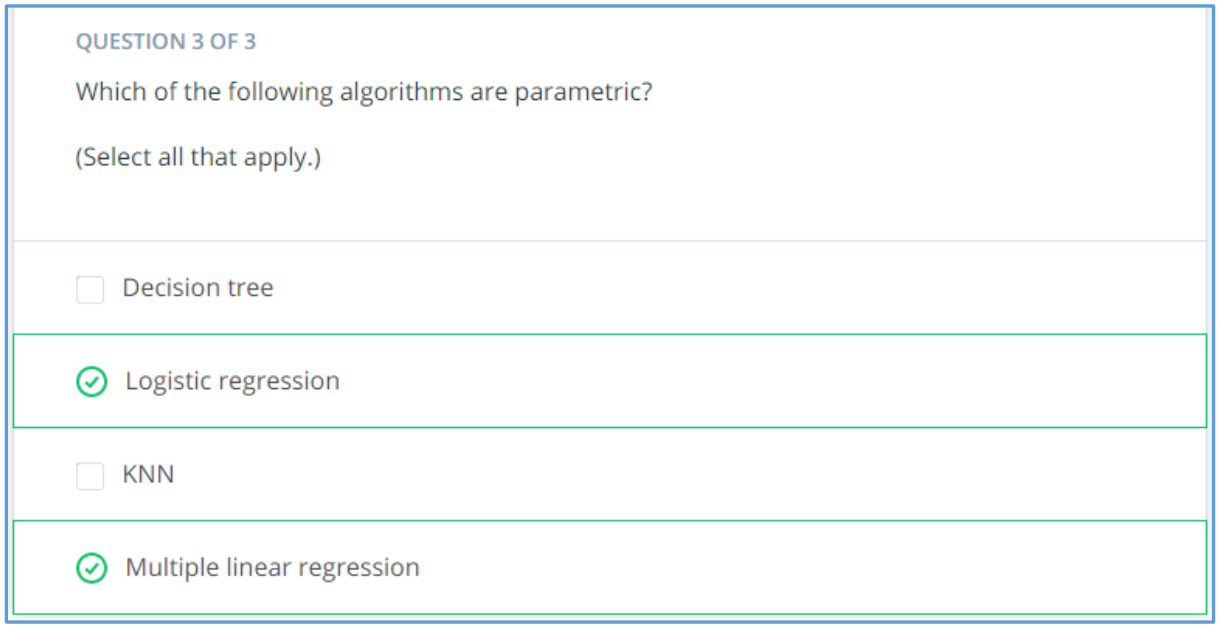
- **High flexibility**, in the sense that they are capable of fitting a large number of functional forms
- **Power** by making weak or no assumptions on the underlying function
- **High performance** in the prediction models that are produced

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







Chapter 27: 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:

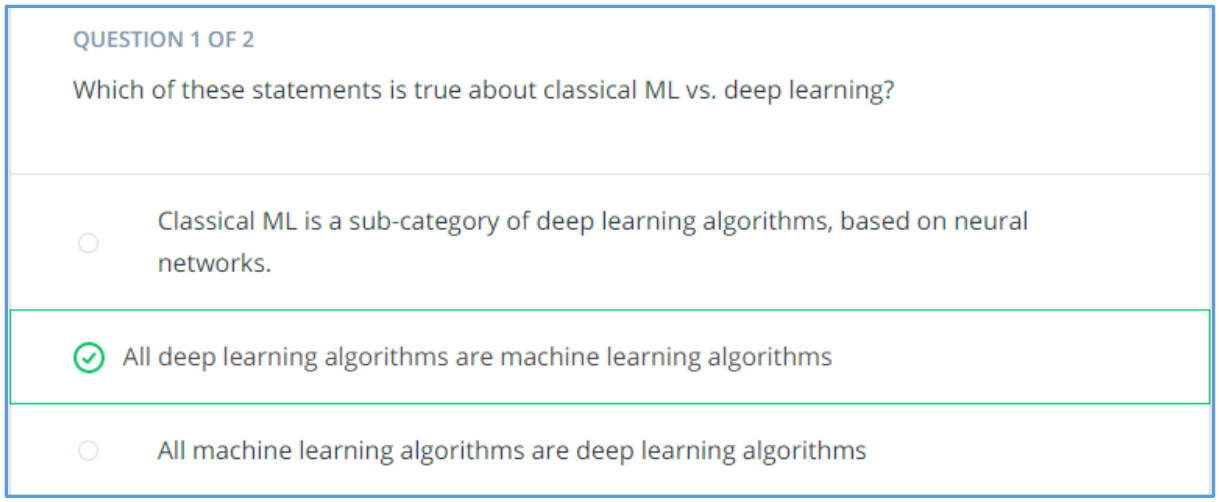
- Difficult to explain trained data
- Require significant computational power

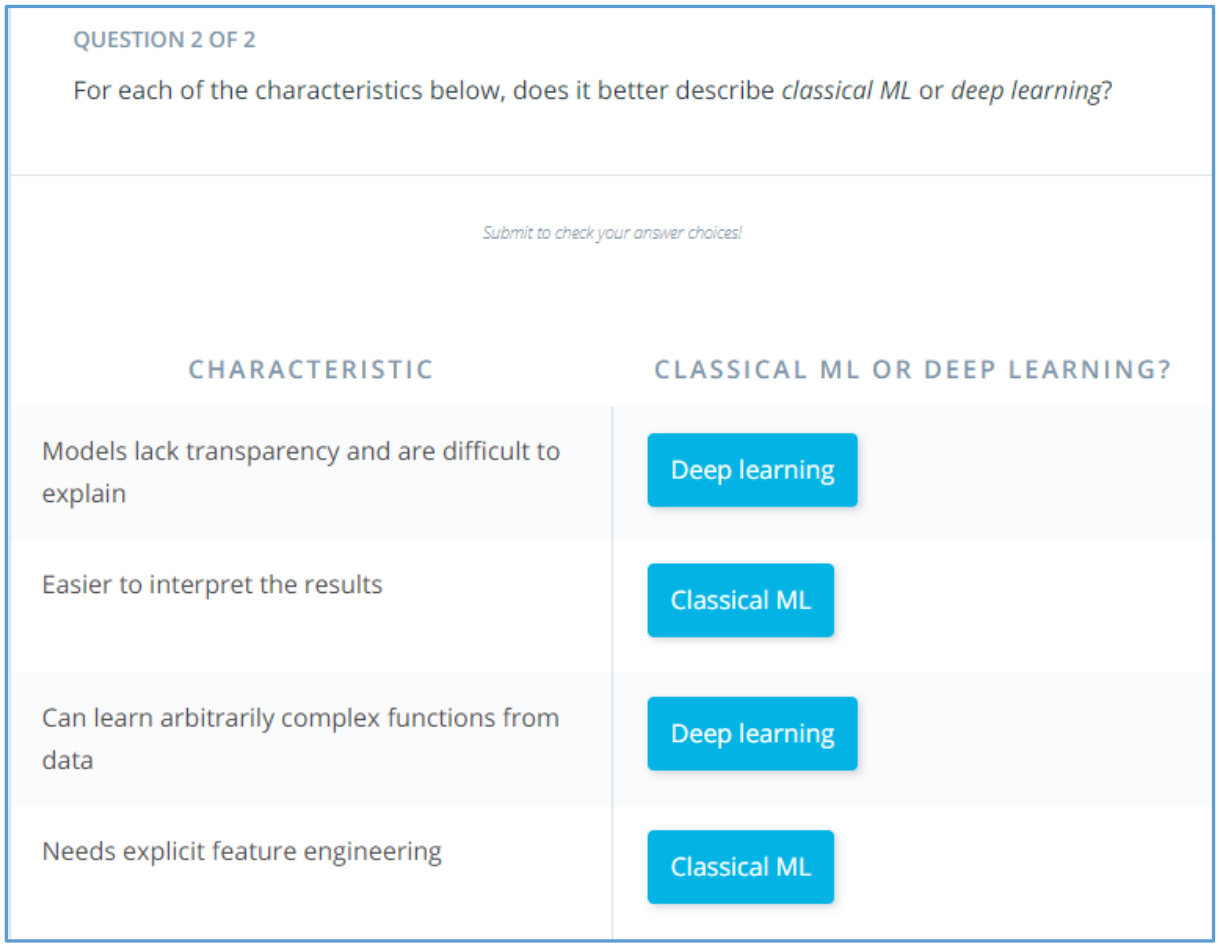
Classical ML advantages:

- 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





Chapter 28: Approaches to ML

There are three main approaches to machine learning:

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

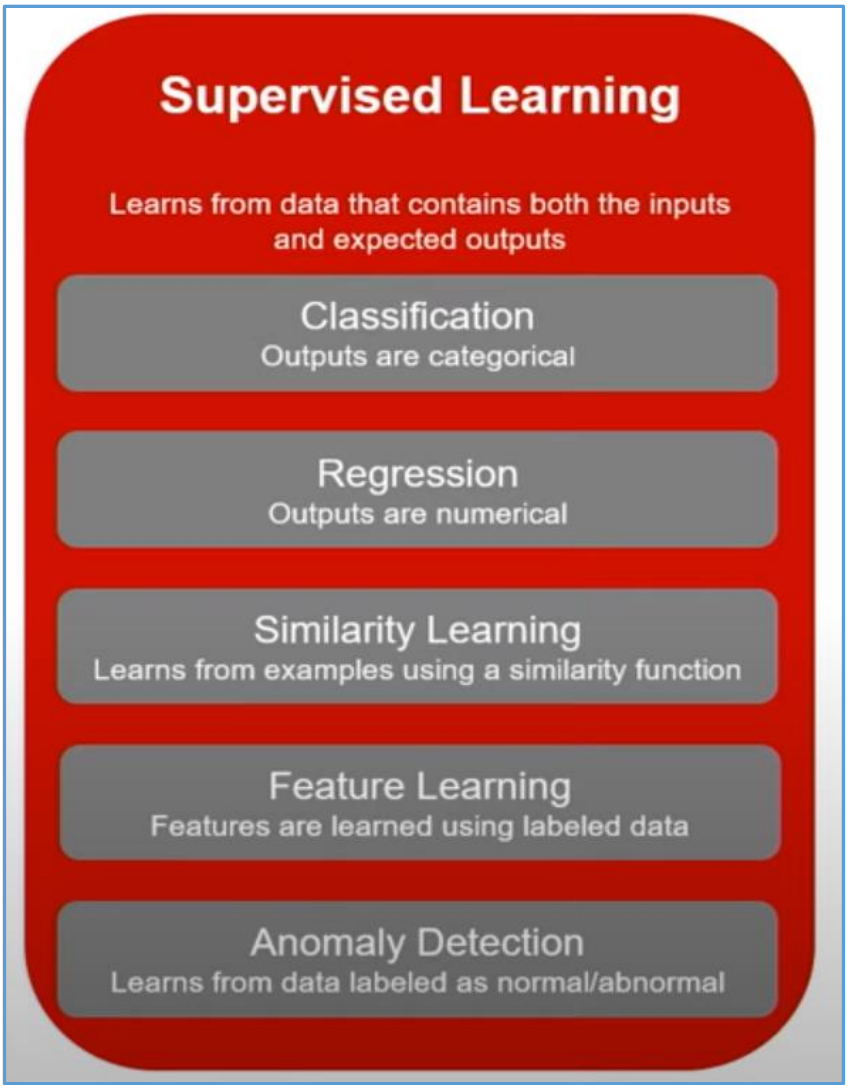
We will take a short, high-level look at these approaches here, and then revisit them in more detail in later lessons.

Supervised learning

Learns from data, that contains both the inputs and expected outputs (e.g., labelled 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 labelled as normal/abnormal.**

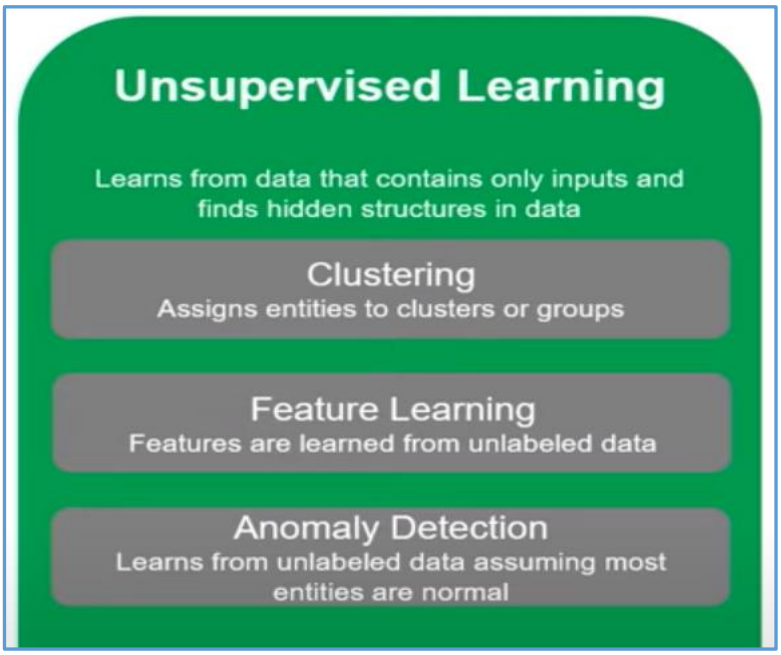


Unsupervised learning

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

- **Clustering: Assigns entities to clusters or groups.**

- **Feature learning**: **Features are, learned from unlabelled data.**
- **Anomaly detection: Learns from unlabelled data, using the assumption that the majority of entities are normal.**

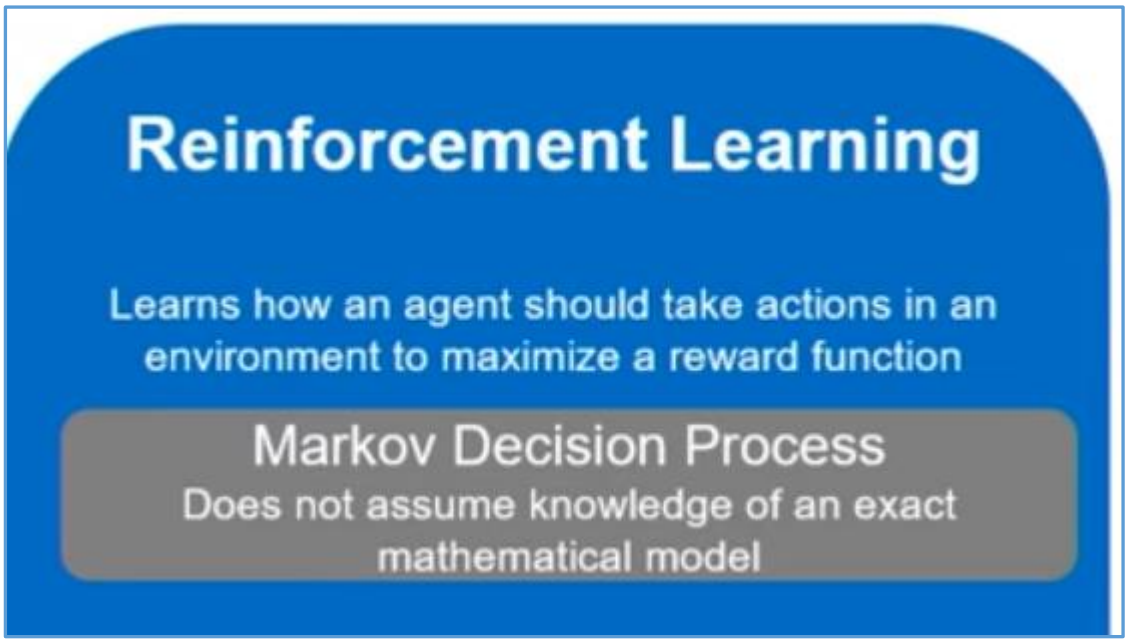


Reinforcement learning

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

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

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

It is true that supervised learning can be an approach to anomaly detection. However, in most, anomaly detection cases, the training data is too highly imbalanced for effectively using supervised learning.

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

Chapter 29: 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. Parametric functions would tend to have higher bias compared to Non-parametric functions**

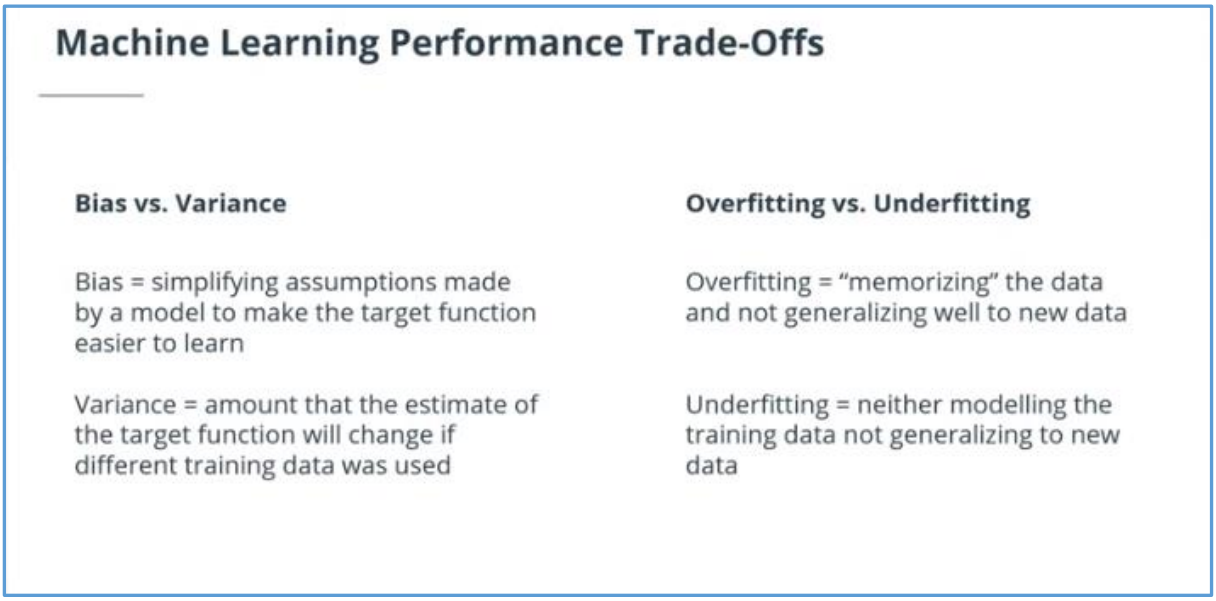
Variance measures how much the target function will *change* if different training data is used. Variance could be, caused by modelling the *random noise* in the training data. **High model complexity tends to have a high variance.**

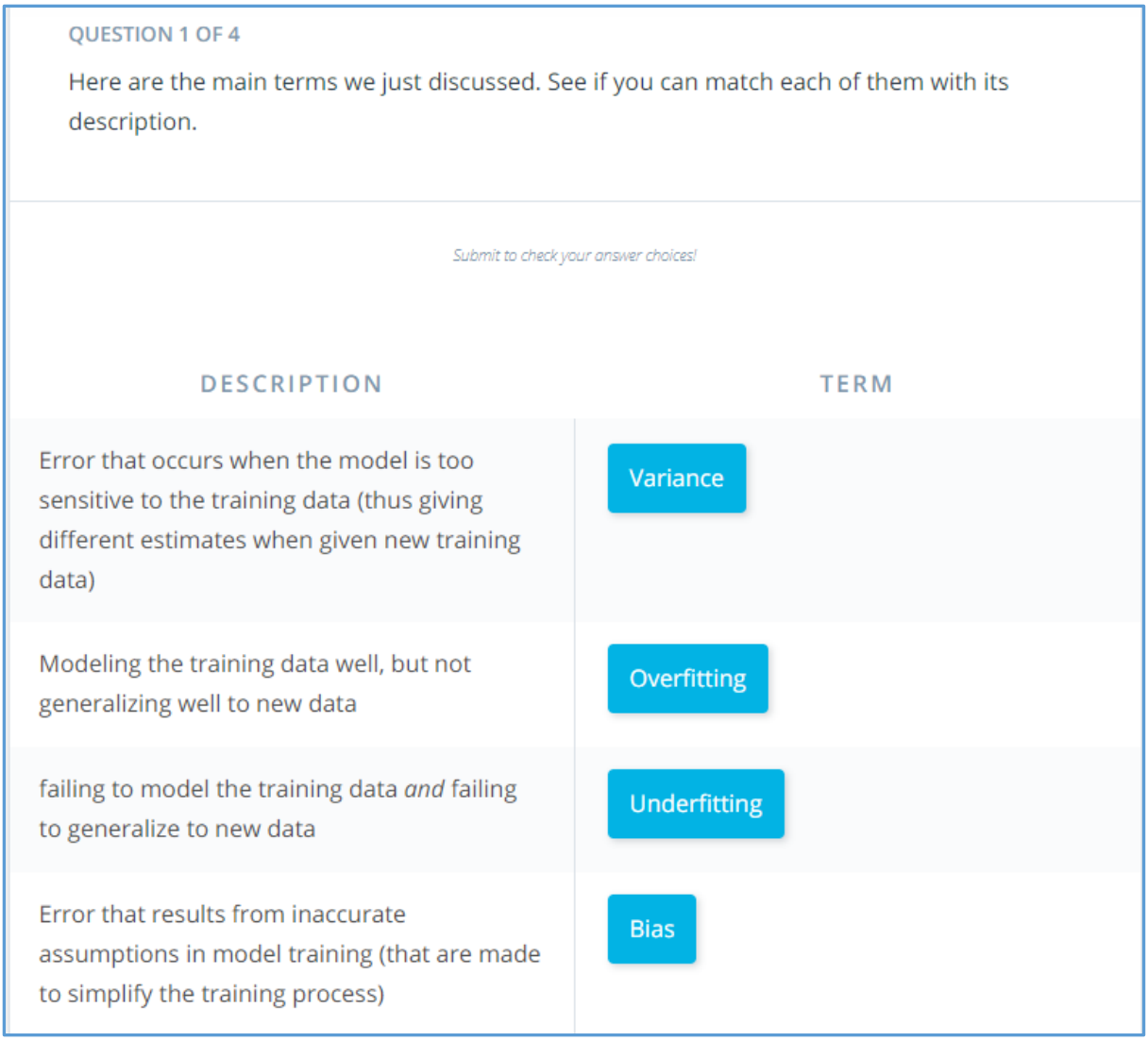
As a general trend, parametric and linear algorithms often have high bias and low variance, whereas non-parametric and non-linear algorithms often have low bias and high variance

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.





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).

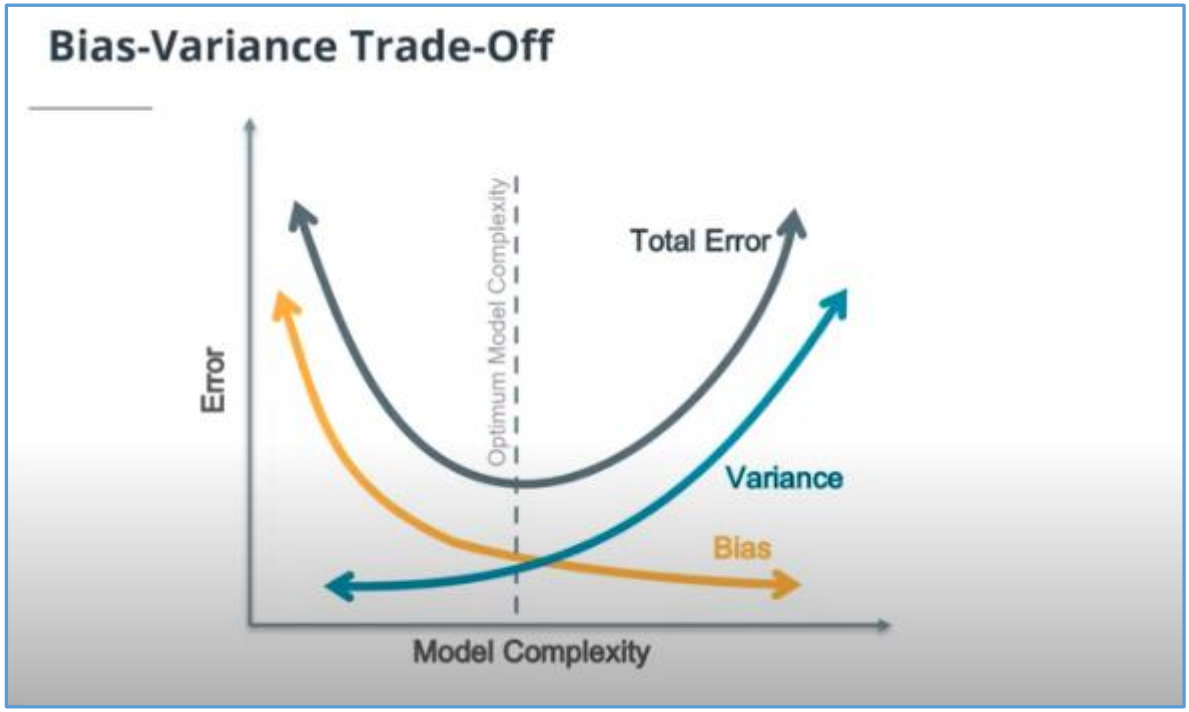
Prediction error = Bias error + Variance error + Irreducible error

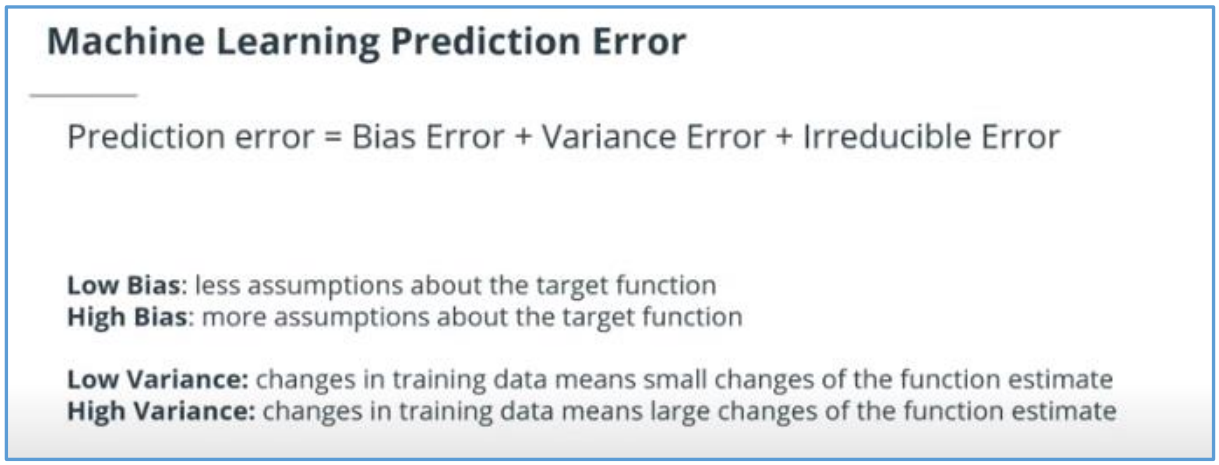
Low bias means *fewer* assumptions about the target function. Some examples of algorithms with low bias are KNN and decision trees. Having fewer assumptions can help generalize relevant relations between features and target outputs.
In contrast, ***high bias*** means *more* assumptions about the target 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. However, 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.



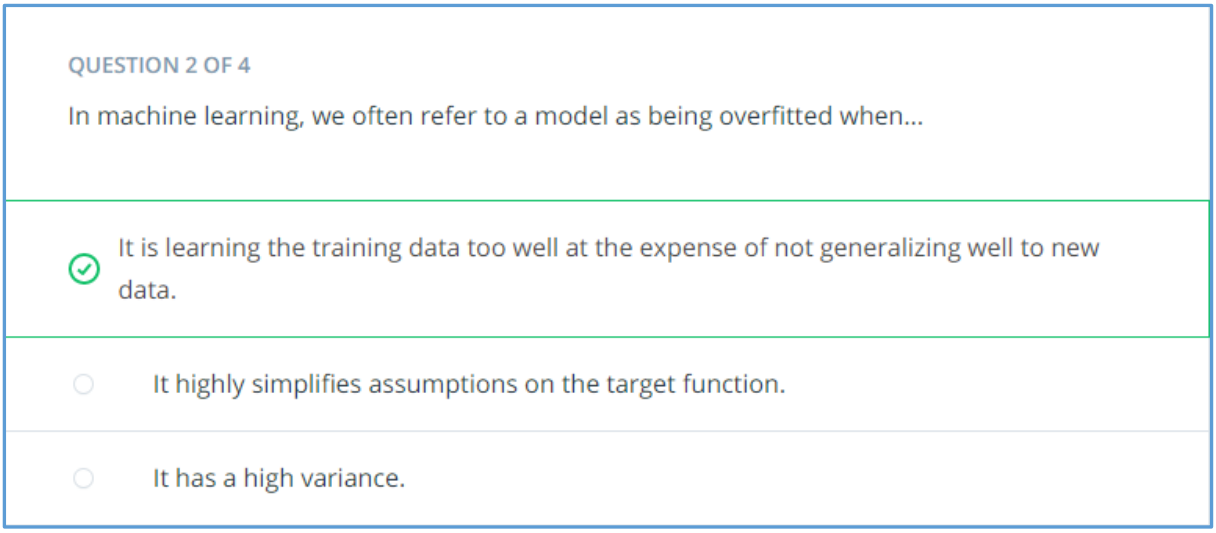


Low Bias >> Less Assumptions >> Non Parametric functions >> Complex Model>> High Variance >> Overfitting

High Bias >> More Assumptions >> Parametric Functions >> Simple Model >> Low Variance >> Underfitting

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 estimate 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 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.

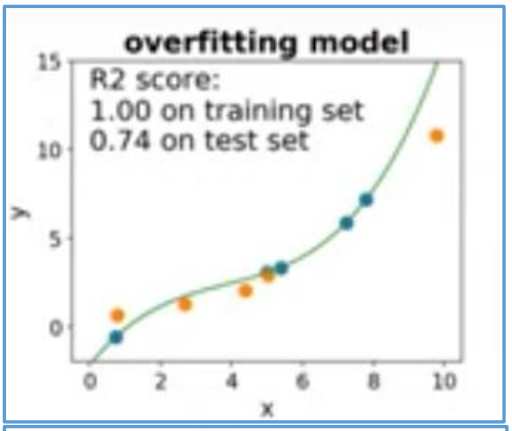
QUESTION 4 OF 4

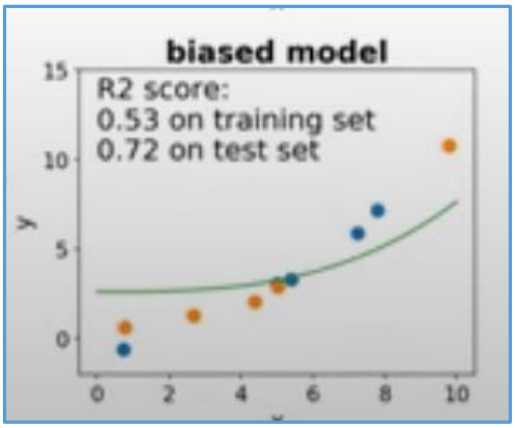
Which technique can be used to reduce overfitting?

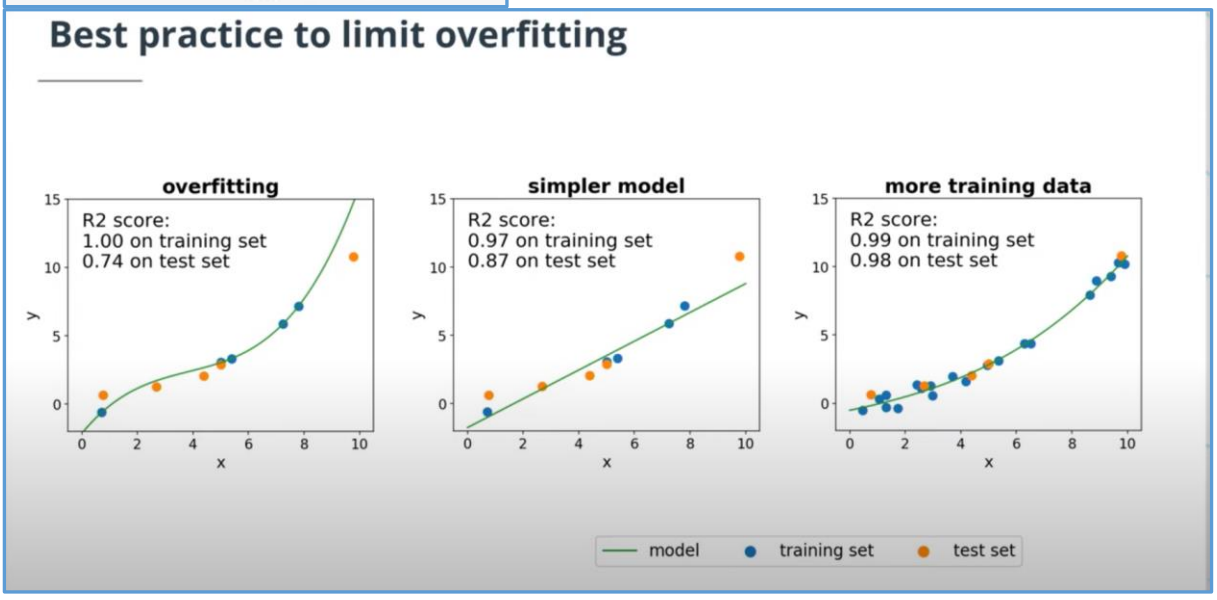
(Select all that apply.)

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

		Model performance on training data	
		Poor	**Good**
Model performance on new data	**Poor**	Underfitting (high bias)	Overfitting (high variance)
	Good	Very unlikely	Optimal







Chapter 30: 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.