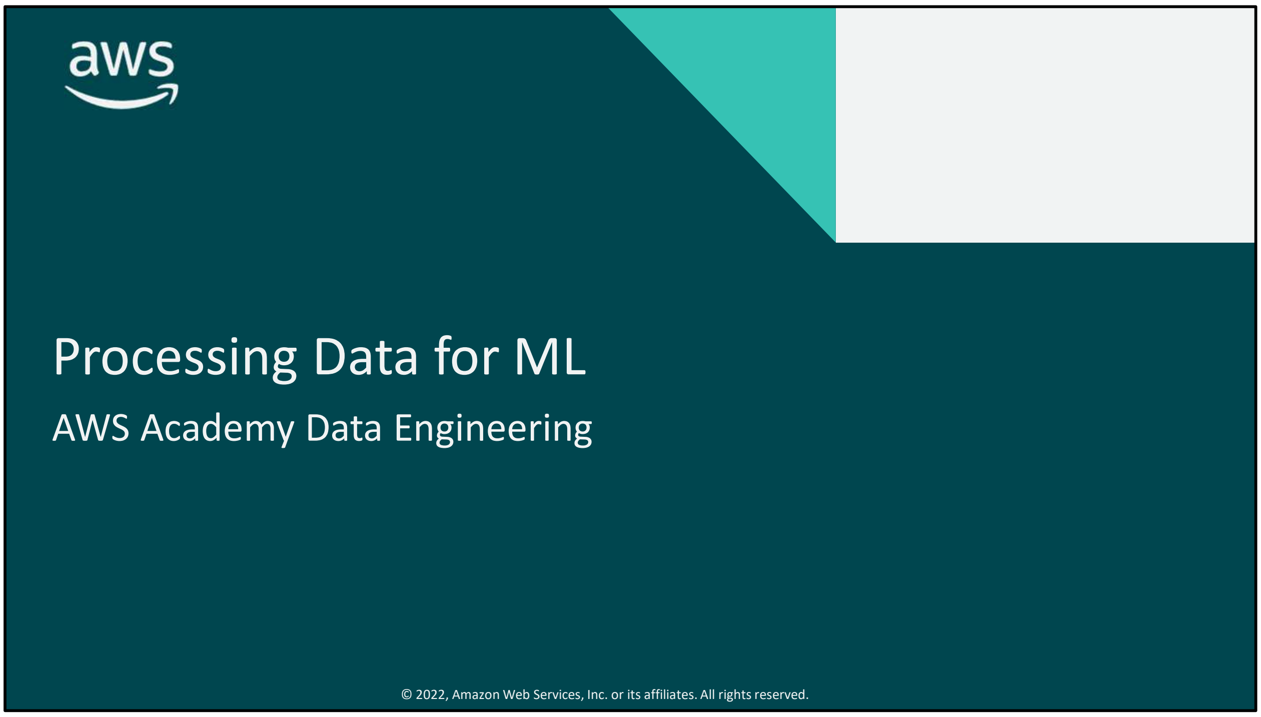


Processing Data for ML

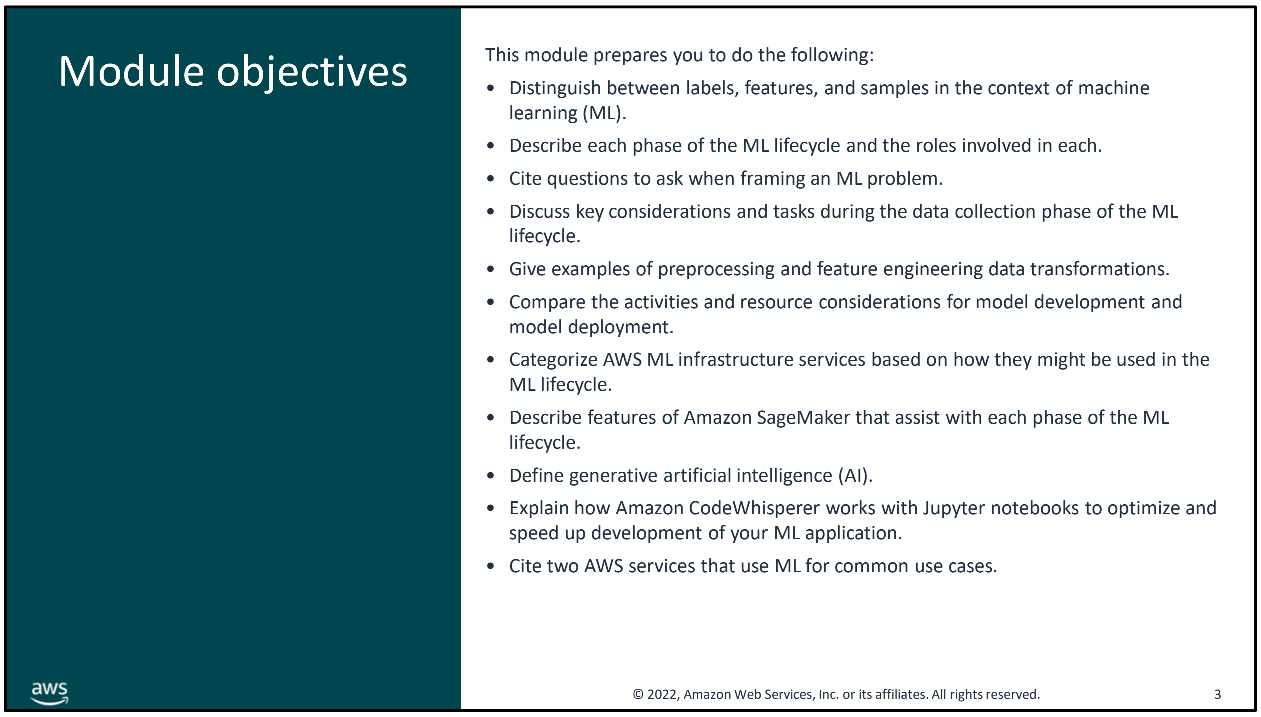
AWS Academy Data Engineering

~labeling activities – ground truth
|Slide number 1
|Instructor notes
|Because this course is targeted to the data engineering role, this module primarily focuses on topics that are related to creating the infrastructure to support machine learning (ML) processing. The first section after the introduction provides a brief overview of ML concepts and terminology. This section could be optional depending on your audience's familiarity with these concepts. This module will discuss topics such as modeling, training, tuning, and deploying at a high level, with a focus on how data is used and moves through the system.
|
|Student notes
Welcome to the Processing Data for ML module.

Introduction

Processing Data for ML

|Slide number 2
|Instructor notes
|
|Student notes
This introduction section describes the content of this module.



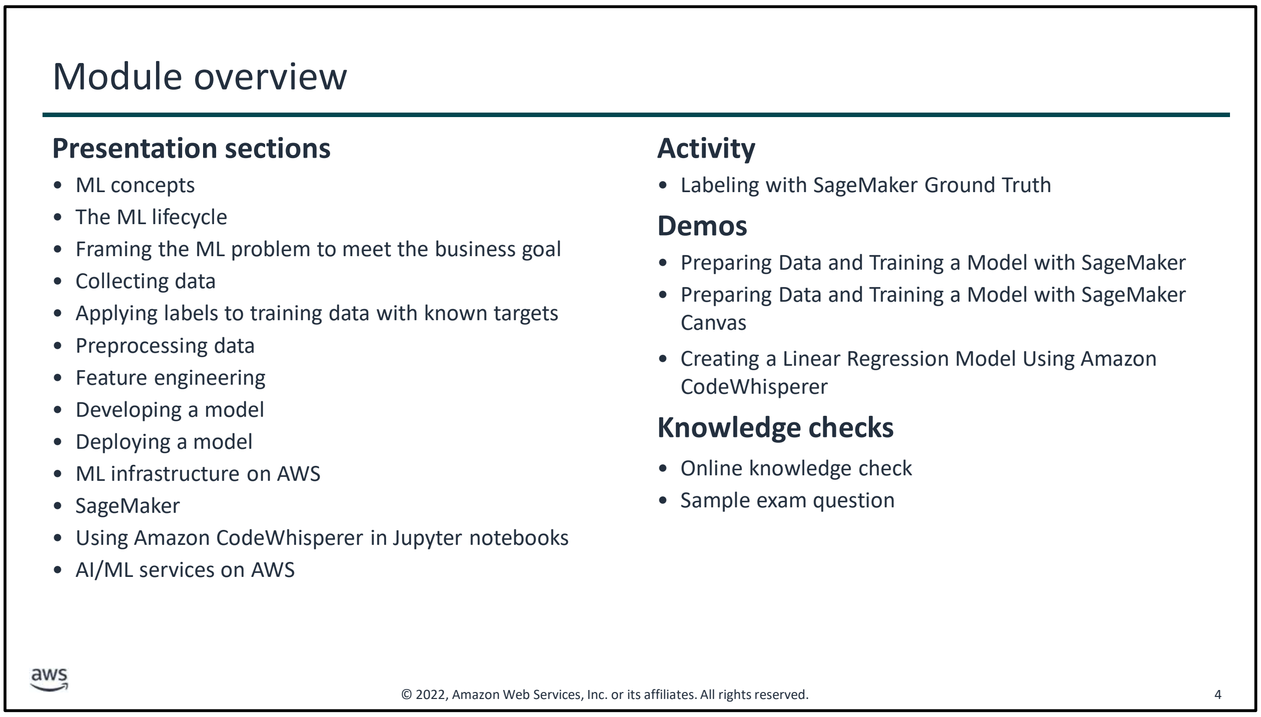
Module objectives

This module prepares you to do the following:

- Distinguish between labels, features, and samples in the context of machine learning (ML).
- Describe each phase of the ML lifecycle and the roles involved in each.
- Cite questions to ask when framing an ML problem.
- Discuss key considerations and tasks during the data collection phase of the ML lifecycle.
- Give examples of preprocessing and feature engineering data transformations.
- Compare the activities and resource considerations for model development and model deployment.
- Categorize AWS ML infrastructure services based on how they might be used in the ML lifecycle.
- Describe features of Amazon SageMaker that assist with each phase of the ML lifecycle.
- Define generative artificial intelligence (AI).
- Explain how Amazon CodeWhisperer works with Jupyter notebooks to optimize and speed up development of your ML application.
- Cite two AWS services that use ML for common use cases.

|Slide number 3
|Instructor notes
|
|Student notes

This module provides a brief look at machine learning (ML) and focuses on the ML lifecycle, which guides how data engineers and data scientists collect and process data for ML models. The module takes a quick look at general ML concepts and describes the activities that are performed in each phase of the ML lifecycle. You will learn about framing the ML problem, collecting and preprocessing data, feature engineering, model development, and model deployment. You will also look at the AWS infrastructure services that you can use to implement the ML lifecycle. You will learn how Amazon CodeWhisperer can help speed up development of your ML application. Finally, you will be introduced to other artificial intelligence (AI) and ML services that you can use to solve common use cases without needing a lot of ML knowledge.


Module overview

Presentation sections

- ML concepts
- The ML lifecycle
- Framing the ML problem to meet the business goal
- Collecting data
- Applying labels to training data with known targets
- Preprocessing data
- Feature engineering
- Developing a model
- Deploying a model
- ML infrastructure on AWS
- SageMaker
- Using Amazon CodeWhisperer in Jupyter notebooks
- AI/ML services on AWS

Activity

- Labeling with SageMaker Ground Truth

Demos

- Preparing Data and Training a Model with SageMaker
- Preparing Data and Training a Model with SageMaker Canvas
- Creating a Linear Regression Model Using Amazon CodeWhisperer

Knowledge checks

- Online knowledge check
- Sample exam question

~ script notes: for digital script, activity and demo information will be removed/skipped
|Slide number 4
|Instructor notes
|Each module has an introduction, content sections, and a wrap-up. The wrap-up for this module contains a sample exam question for you to review with the students.
|
|Student notes
The objectives of this module are presented across multiple sections.

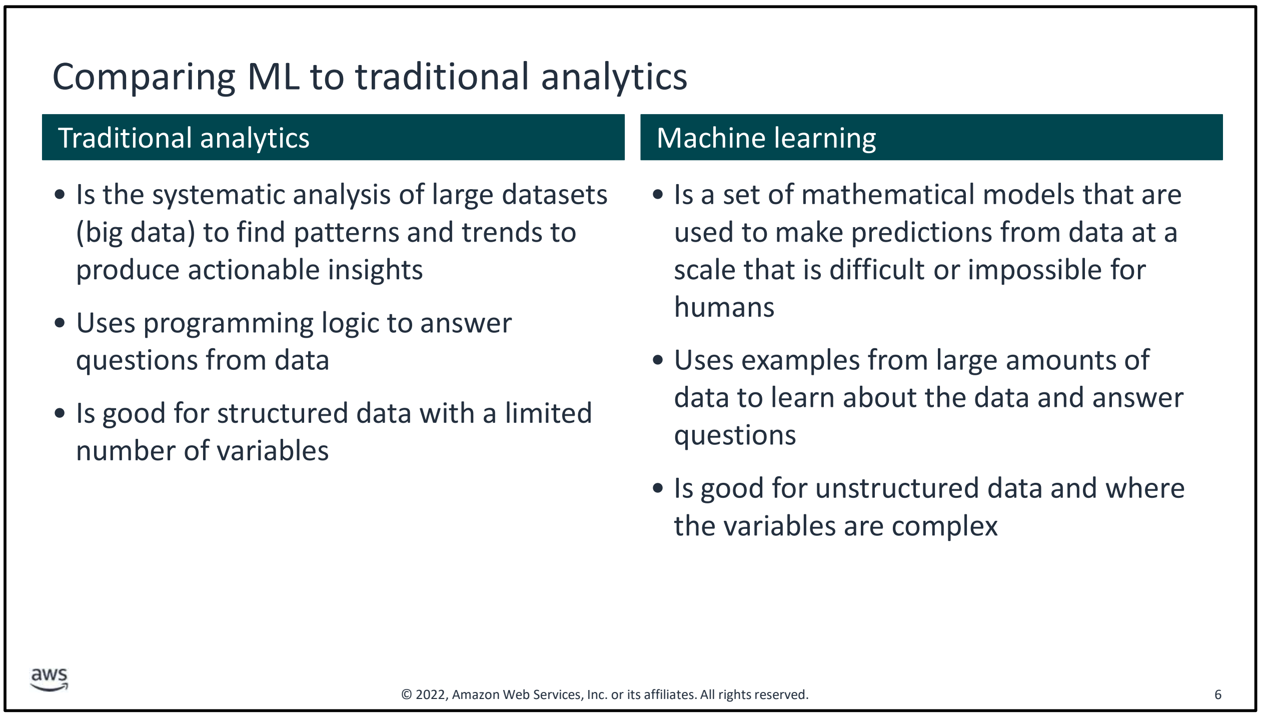
You will also complete a hands-on activity that uses Amazon SageMaker Ground Truth to label images, view two video demonstrations about Amazon SageMaker and one video demonstration about Amazon CodeWhisperer.

The module wraps up with an online knowledge check and a sample exam question that covers the presented material.

ML concepts

Processing Data for ML

|Slide number 5
|Instructor notes
|This section provides brief coverage of ML topics. Depending on your audience, you might skip it completely, or you might find that you need to go deeper on these topics.
|
|Student notes
This section introduces some foundational concepts about machine learning that will help to provide context to the discussion of data processing, which is the focus of this module.



Comparing ML to traditional analytics

Traditional analytics

- Is the systematic analysis of large datasets (big data) to find patterns and trends to produce actionable insights
- Uses programming logic to answer questions from data
- Is good for structured data with a limited number of variables

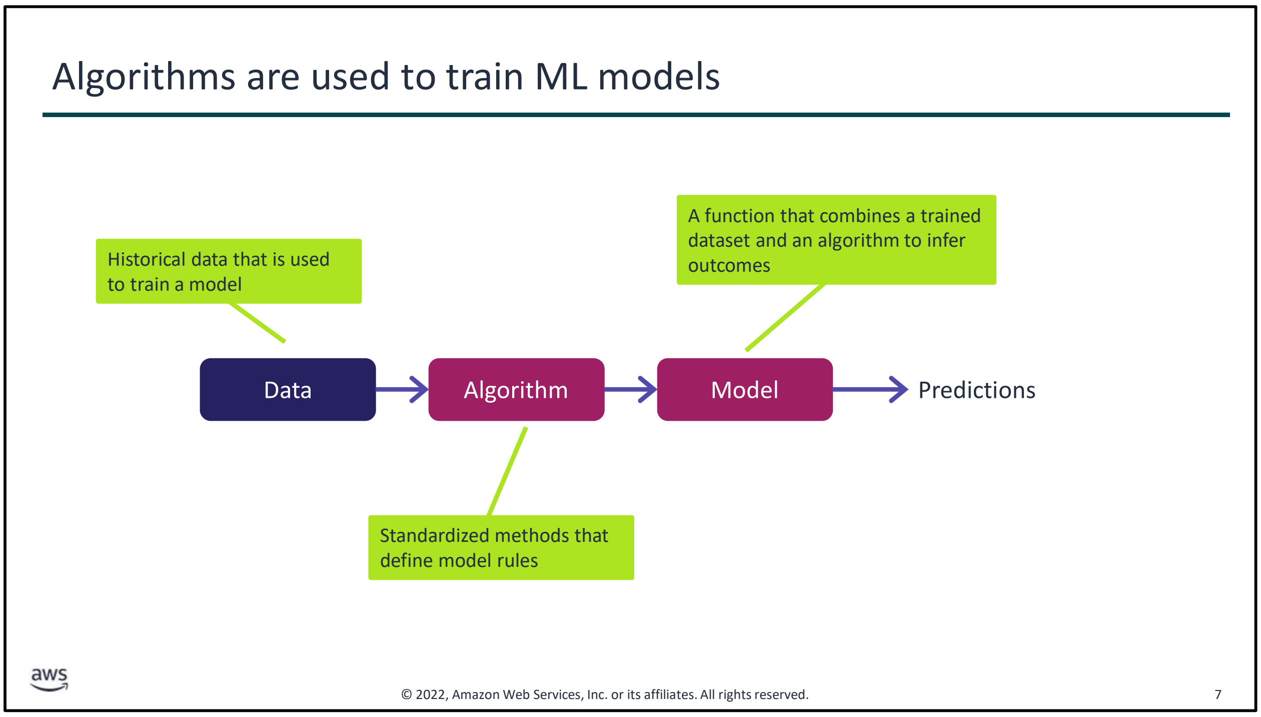
Machine learning

- Is a set of mathematical models that are used to make predictions from data at a scale that is difficult or impossible for humans
- Uses examples from large amounts of data to learn about the data and answer questions
- Is good for unstructured data and where the variables are complex

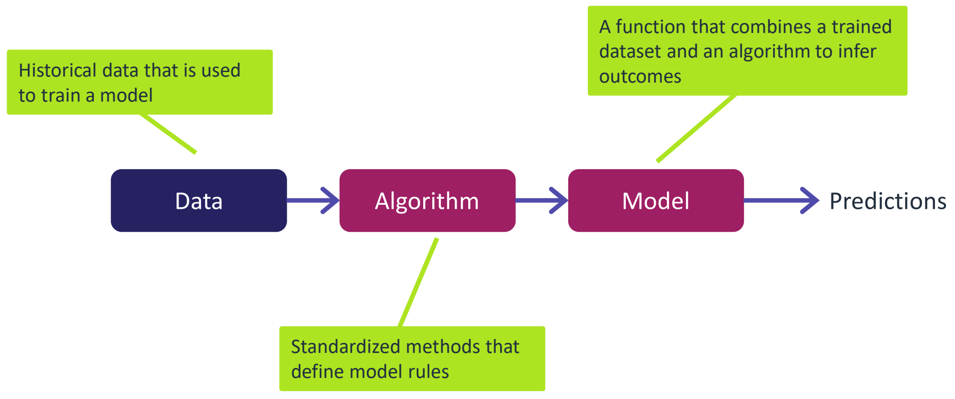
|Slide number 6
|Instructor notes
|This slide was shown in the Data-Driven Organizations module to introduce the distinction between traditional analytics and ML.
|
|Student notes
The Data-Driven Organizations module presents this contrast between the characteristics of traditional analytics and the field of AI/ML.

With traditional methods, developers or data scientists write programming logic to represent a set of rules, which are applied to data to get outcomes.

With machine learning, the rules are not hardcoded. ML practitioners build a mathematical function that finds patterns in historical data and uses the patterns to learn how to predict outcomes when given new data.



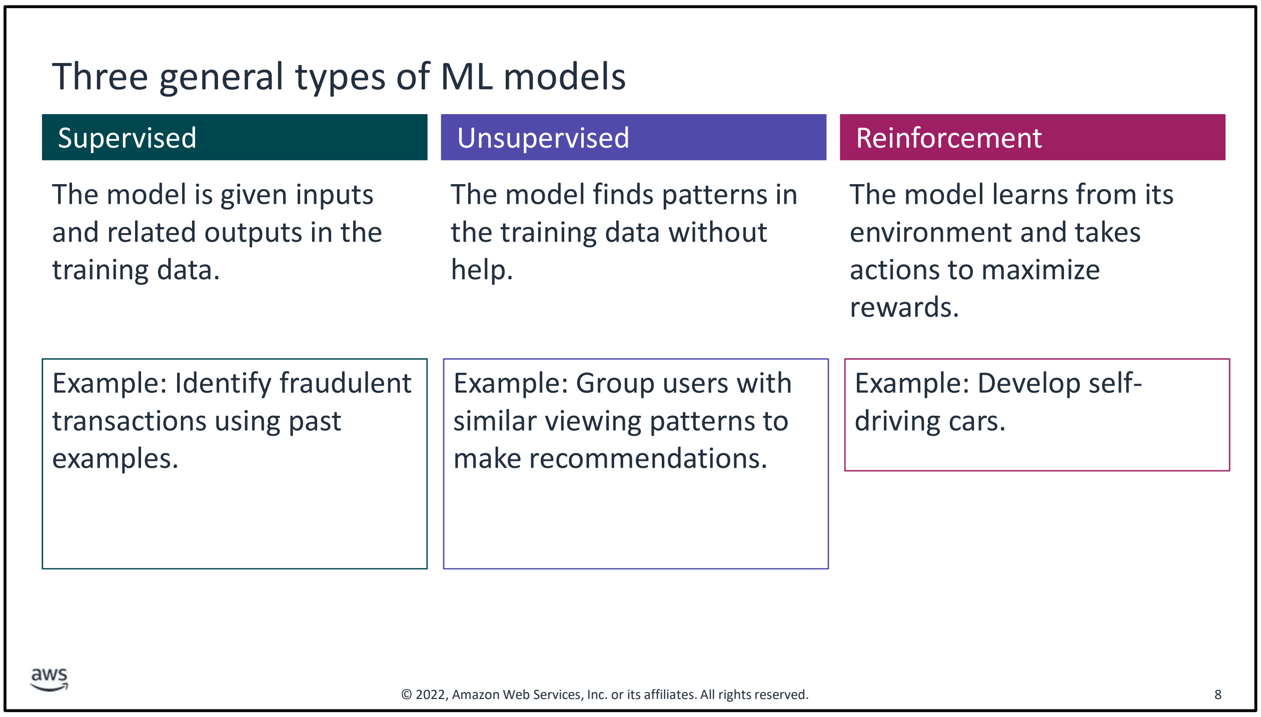
Algorithms are used to train ML models


Historical data that is used to train a model
A function that combines a trained dataset and an algorithm to infer outcomes
Data
Algorithm
Model
Predictions
Standardized methods that define model rules

|Slide number 7
|Instructor notes
|
|Student notes

An ML model is a mathematical function that maps inputs to a set of predicted outcomes by using algorithms. When you start to build an ML model, the precise definition of the function isn’t yet known. You start with a clean dataset that has been prepared for ML, and an algorithm that fits the data and the type of business problem that you are trying to solve. Algorithms are standardized methods that define the rules of the model. You might think of the algorithm as a recipe and the model as your cake. You use the algorithm with your ingredients (data) and evaluate what it produces (how does the cake taste?) and then adjust.

You could use the same ingredients and a different recipe, and that might produce a slightly different cake—or an entirely different dessert. It's the combination of the ingredients and the recipe that define the end product. The model is the combination of your trained dataset and the algorithm that you use.



Three general types of ML models

Supervised	Unsupervised	Reinforcement
The model is given inputs and related outputs in the training data.	The model finds patterns in the training data without help.	The model learns from its environment and takes actions to maximize rewards.
Example: Identify fraudulent transactions using past examples.	Example: Group users with similar viewing patterns to make recommendations.	Example: Develop self-driving cars.

~derived from machine learning pipeline course

|Slide number 8

|Instructor notes

|

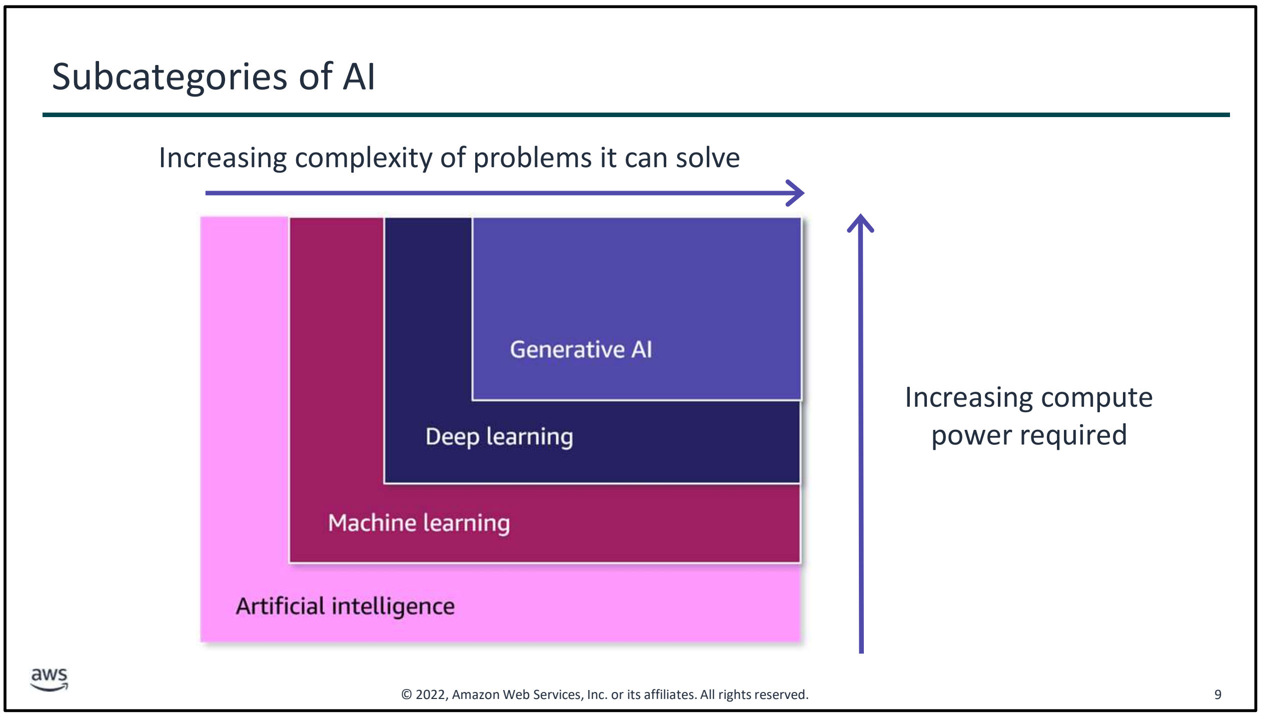
|Student notes

There are three general types of ML models. The processing steps are different for each of them, but they all share the characteristics of using a dataset and an algorithm to train a model to produce predictions.

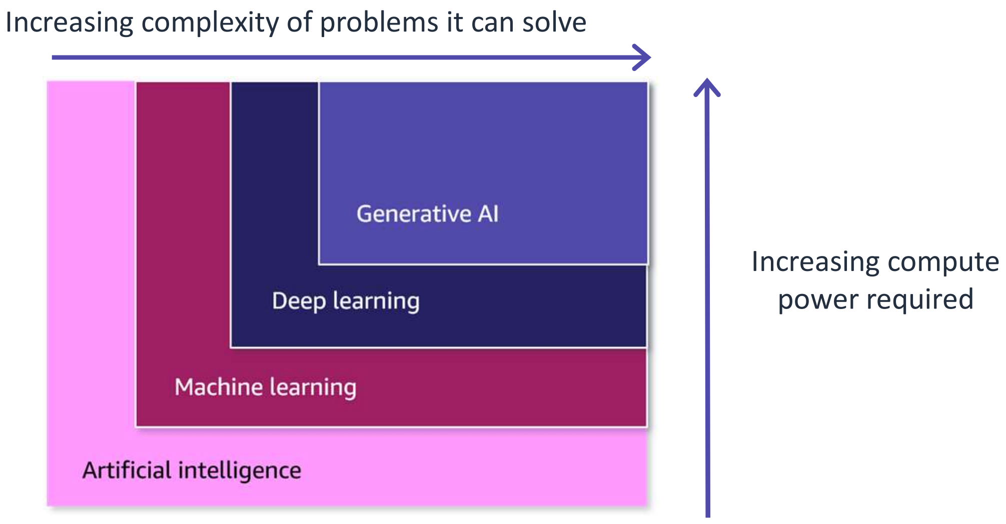
In *supervised learning*, a model uses known inputs and outputs to generalize future outputs. The examples throughout this module are based on supervised learning, which is currently the most common approach in use. Examples include identifying which transactions appear to be fraudulent (Is this fraud? Yes or no) and categorizing objects into a group (this image contains a dog, cat, or bear).

In *unsupervised learning*, the model doesn't know inputs or outputs from the historical data, so the model finds patterns in the data without help. Unsupervised learning is used when you need to identify clusters of related elements within the data. An example is grouping users with similar viewing patterns.

With *reinforcement learning*, the model interacts with its environment and learns through trial and error to take actions that will maximize rewards. Self-driving cars are an example of reinforcement learning. AWS DeepRacer provides a hands-on way to learn about reinforcement learning. The service provides a 1/18th scale racecar that is driven by reinforcement learning. Anyone who is interested can build models and see their results. The link to the AWS DeepRacer overview page is included in the course resources.



Subcategories of AI


Increasing complexity of problems it can solve
Generative AI
Deep learning
Machine learning
Artificial intelligence
Increasing compute power required

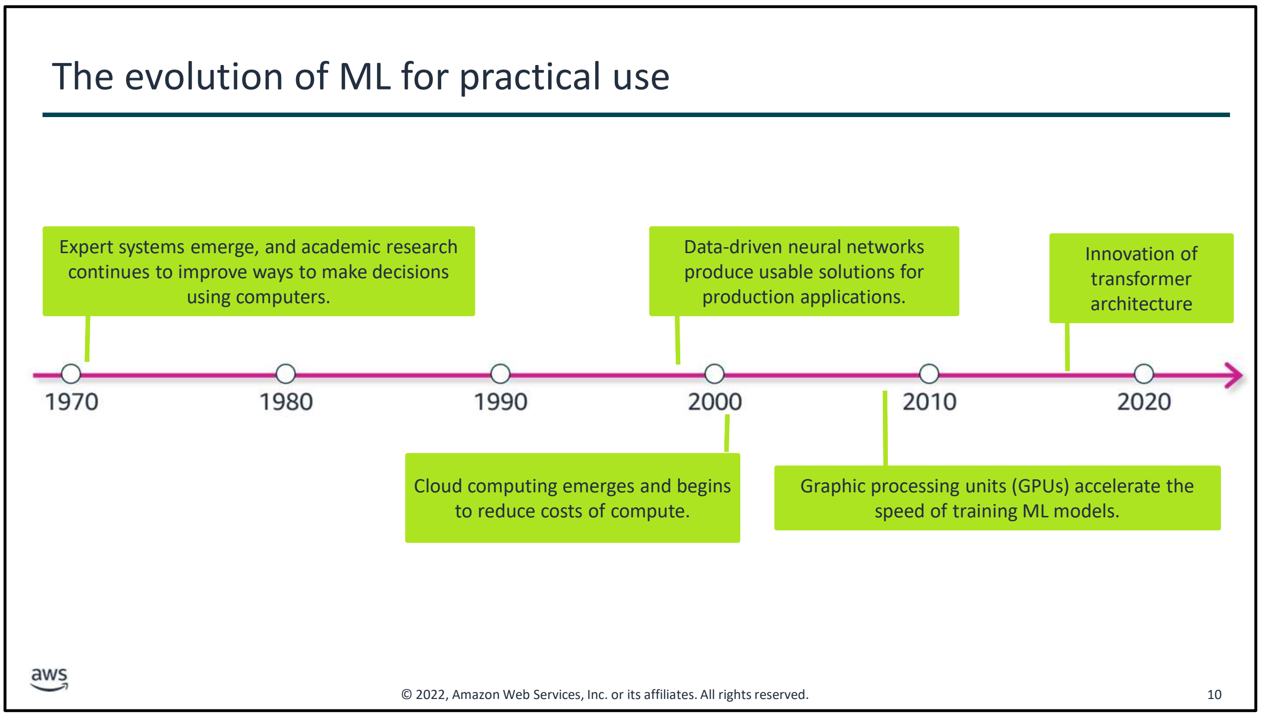
|Slide number 9
|Instructor notes
|
|Student notes

The Data-Driven Organizations module introduces ML as a subcategory of AI. Deep learning is a subcategory of ML whose algorithms use neural networks. Generative AI is a subcategory of deep learning.

A *neural network* is based on a collection of connected units or nodes that loosely models the neurons in a biological brain. With deep learning, a model can recognize patterns that are several magnitudes more complex than other ML models could. A deep learning example is a machine that can learn to play chess by playing against itself.

Deep learning can solve more complex problems but requires more compute power than other methods of ML. The idea of deep learning and neural networks has been around for decades, but the computing power that is needed to support it has only recently been readily available.

Generative AI is a type of AI that can create new content and ideas, including conversations, stories, images, videos, and music. Generative AI is powered by very large ML models that are pre-trained on vast amounts of data and commonly referred to as foundation models (FMs). Unlike traditional ML models that are single task-oriented and require training, a single FM can be use for different tasks such as text generation or summarization without further training. You can use generative AI to improve customer experience through capabilities such as chatbots, virtual assistants, intelligent contact centers, personalization, and content moderation.

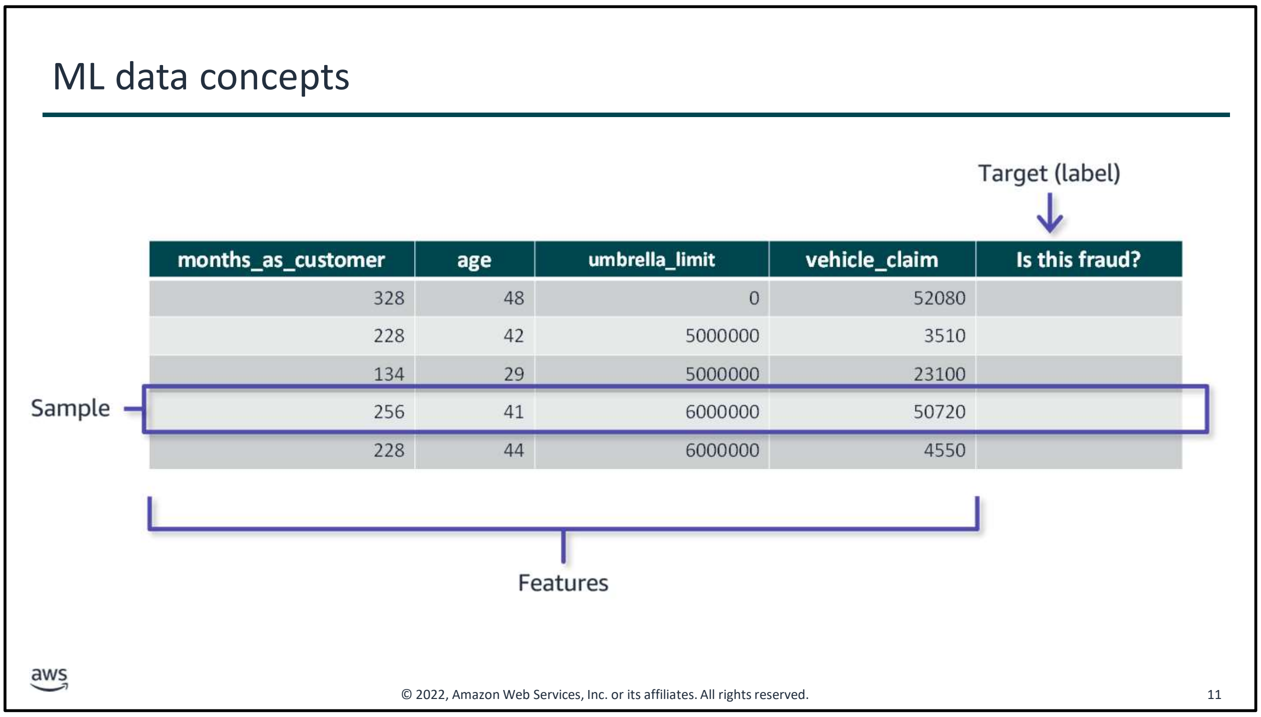


The evolution of ML for practical use

Expert systems emerge, and academic research continues to improve ways to make decisions using computers.
Data-driven neural networks produce usable solutions for production applications.
Innovation of transformer architecture
1970
1980
1990
2000
2010
2020
Cloud computing emerges and begins to reduce costs of compute.
Graphic processing units (GPUs) accelerate the speed of training ML models.

|Slide number 10
|Instructor notes
|
|Student notes
Even in the 1950s, computer experts thought they could build intelligent systems, but computers weren’t fast enough. Then, around the mid 1970s, researchers began using rule-based systems, known as *expert systems*. This approach lasted for approximately 30 years. With these expert systems, academics got a good understanding of what computer-based intelligence actually was, but no production-ready application resulted from this work. However, the evolution of neural network technologies continued until around 1998, when it became possible to start using data to make practical predictions by using what is now called machine learning.

However, to implement these new approaches, the technologies needed to store very large amounts of data inexpensively. These approaches also required large amounts of computing power and a way to transfer data with very low latency between storage and compute. These capabilities didn’t exist in a widely usable way until around 2007. The combination of more affordable cloud-based resources, coupled with the availability of graphic processing units (GPUs), really pushed the machine learning field into its modern era. GPUs were originally developed to power gaming applications but soon became an important infrastructure component for ML solutions. By using GPUs, models can train at greatly accelerated processing speeds, which makes it more practical to use ML solutions for a wider variety of business problems. In 2017, a breakthrough in neural network architecture called the transformer architecture helped make modern day generative AI possible.



ML data concepts

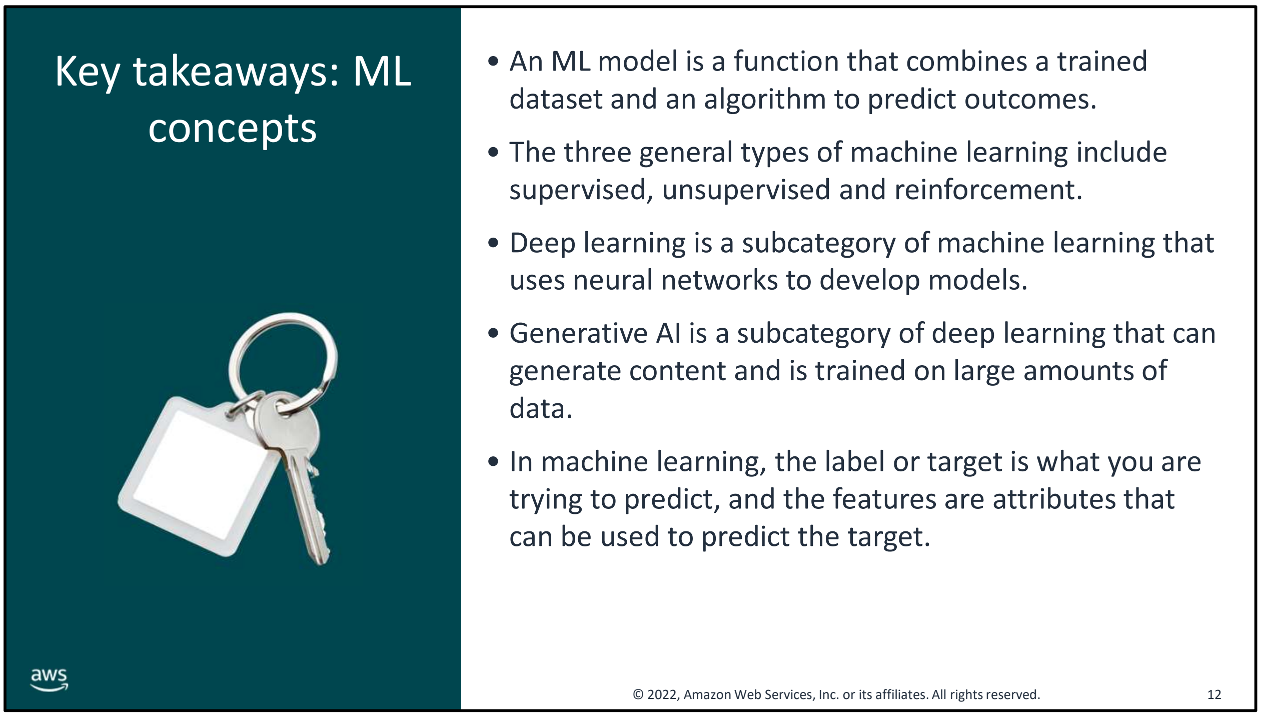
months_as_customer	age	umbrella_limit	vehicle_claim	Is this fraud?
328	48	0	52080	
228	42	5000000	3510	
134	29	5000000	23100	
256	41	6000000	50720	
228	44	6000000	4550	

|Slide number 11
|Instructor notes
|The tabular examples in this module use data that reflects the data that is used in the SageMaker demos that are provided toward the end of the module.
|
|Student notes
This slide uses a tabular data example to depict a few important concepts about the data that is used in ML models. A tabular data example makes it easy to see the components in familiar constructs. Labels and features appear as columns, and samples are represented by rows.

A *label*, or *target*, is the outcome. It’s also known as a *class*, *dependent variable*, or *response*. This is what you are trying to predict.

A *feature* is an attribute or independent variable. It is also known as the *predictor*. Think of these as details that describe the target.

And a *sample* is an occurrence of a target and its features.



Key takeaways: ML concepts

- An ML model is a function that combines a trained dataset and an algorithm to predict outcomes.
- The three general types of machine learning include supervised, unsupervised and reinforcement.
- Deep learning is a subcategory of machine learning that uses neural networks to develop models.
- Generative AI is a subcategory of deep learning that can generate content and is trained on large amounts of data.
- In machine learning, the label or target is what you are trying to predict, and the features are attributes that can be used to predict the target.

|Slide number 12
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

An ML model is a function that uses an algorithm and a trained dataset to predict outcomes.

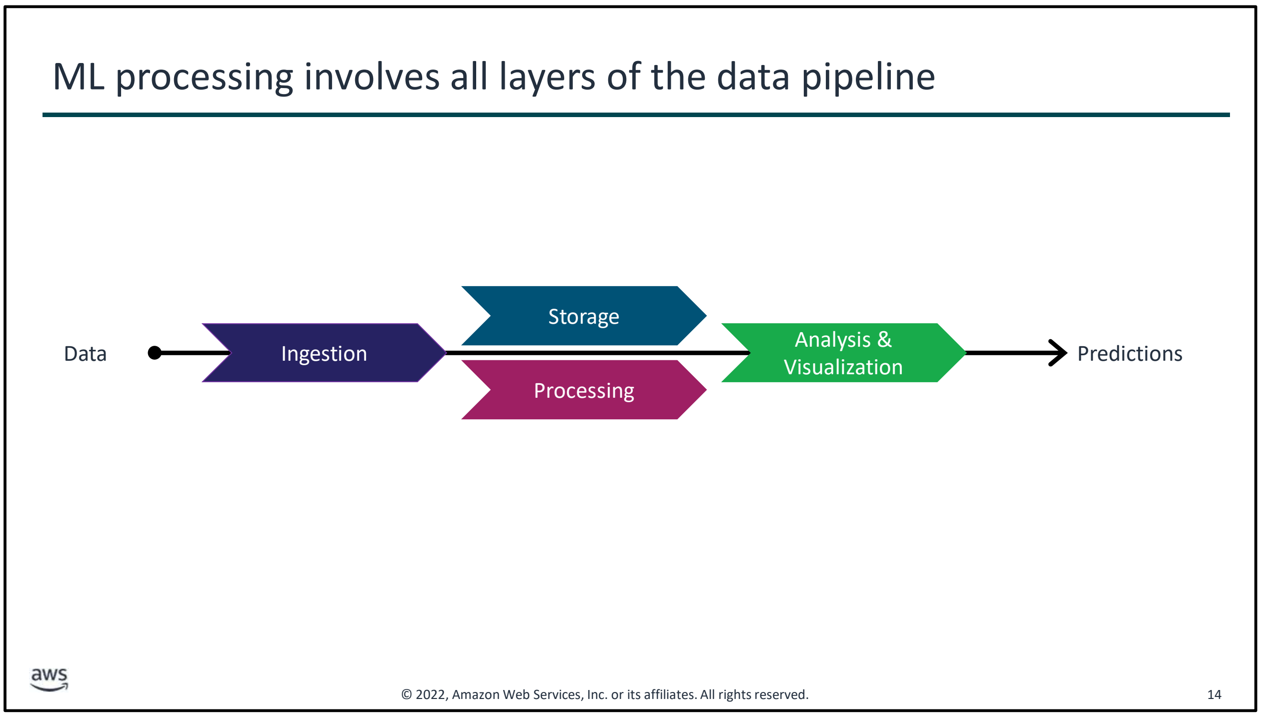
Machine learning can be divided into three types of models: supervised, unsupervised, and reinforcement learning.

Deep Learning is a subcategory within machine learning where the models are developed on a neural network. This enables the models to recognize patterns that are much more complex than with traditional algorithms.

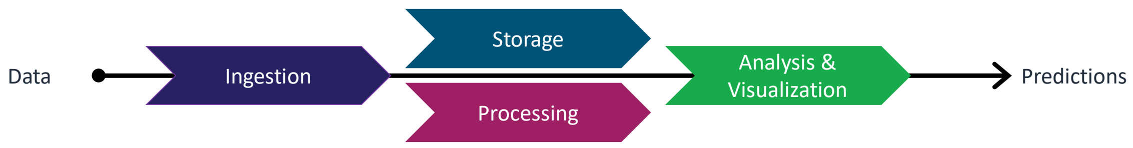
The ML lifecycle

Processing Data for ML

|Slide number 13
|Instructor notes
|The concepts that are presented in the lifecycle sections are drawn primarily from the Machine Learning Lens of the AWS Well-Architected Framework. Because this course focuses on the data engineer role, the course has limited depth around the model building and deployment phases. The module doesn't cover the details of how each phase is implemented. Depending on your audience, you might want to dive into a particular phase or drill down into one of the pillars in more detail. The link to the Machine Learning Lens is provided in the course resources.
|
|Student notes
This section describes a set of general steps that make up the ML lifecycle, which can be used to develop and deploy an ML solution.

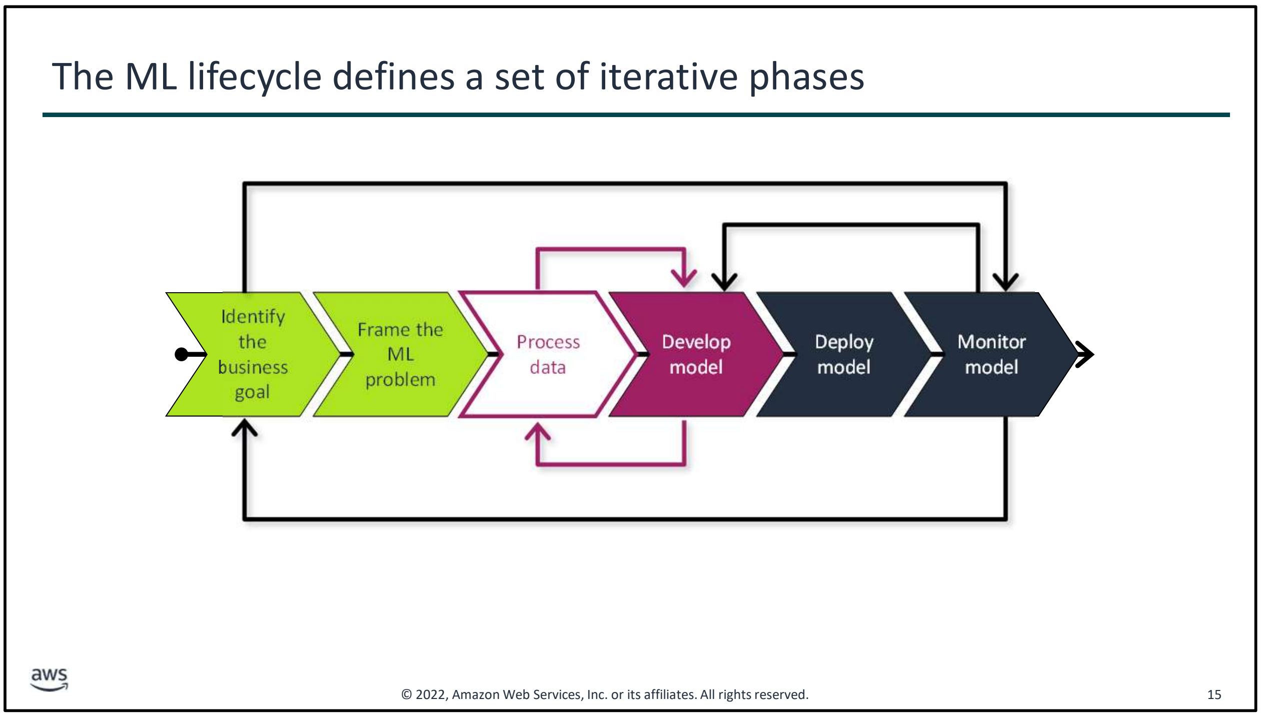


ML processing involves all layers of the data pipeline

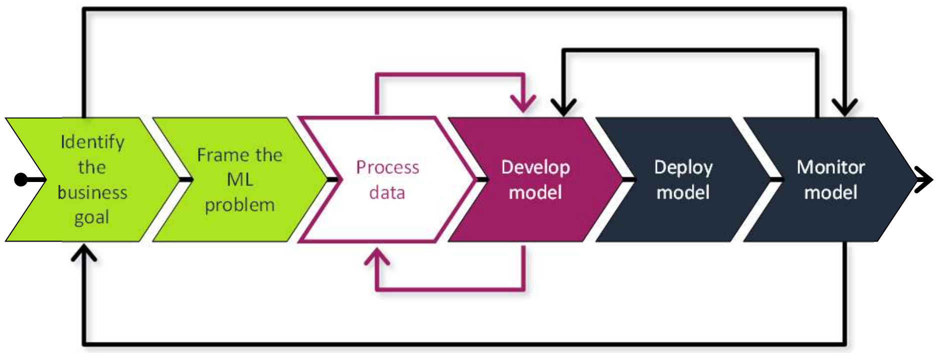


|Slide number 14
|Instructor notes
|
|Student notes

An ML application is a specific use case of a data pipeline. An ML solution will incorporate several types of ingestion, storage, processing, and analysis and visualization activities across the ML lifecycle.



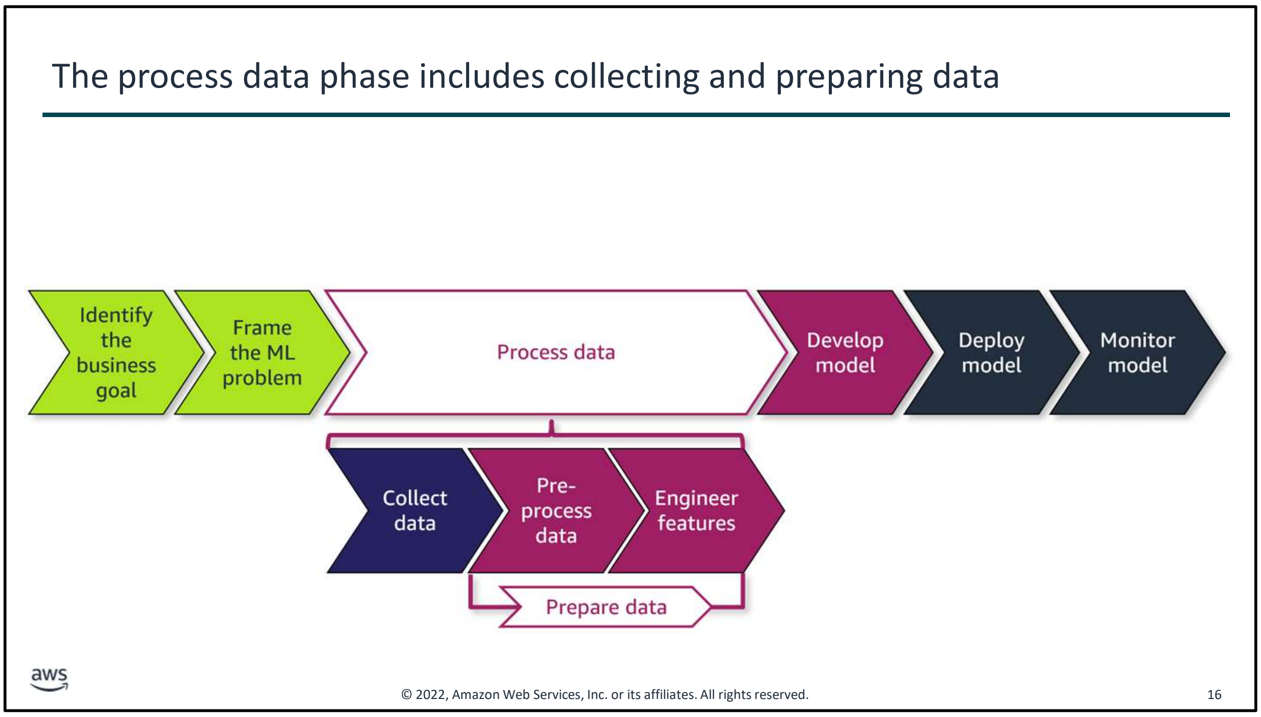
The ML lifecycle defines a set of iterative phases



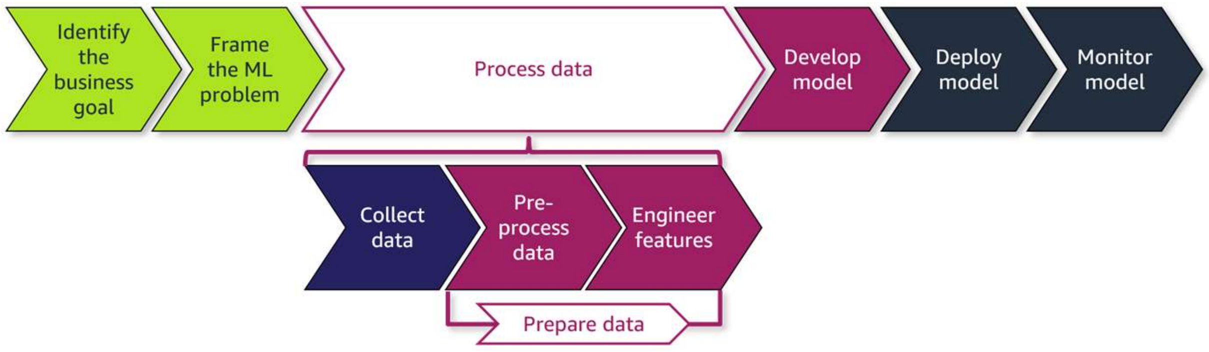
|Slide number 15
|Instructor notes
|
|Student notes

You won’t be surprised to hear that the lifecycle that is depicted on this slide is iterative, and the phases in it aren’t as well defined as the diagram suggests. The color selections on the steps segment the lifecycle into three main sections. The first two phases are about analyzing the problem that you want to solve and whether an ML solution can solve it. The third and fourth phases of the lifecycle are where data processing and modeling occur. And the last two phases are where the ML solution becomes operational.

Note how the business goal drives the process and directly influences the metrics that you should choose to measure and monitor in the deployed model. The results then feed back in to determine whether the business goal is being met and also feed back into model development. Checking that the model is serving the business need and making high-quality predictions is an ongoing process.

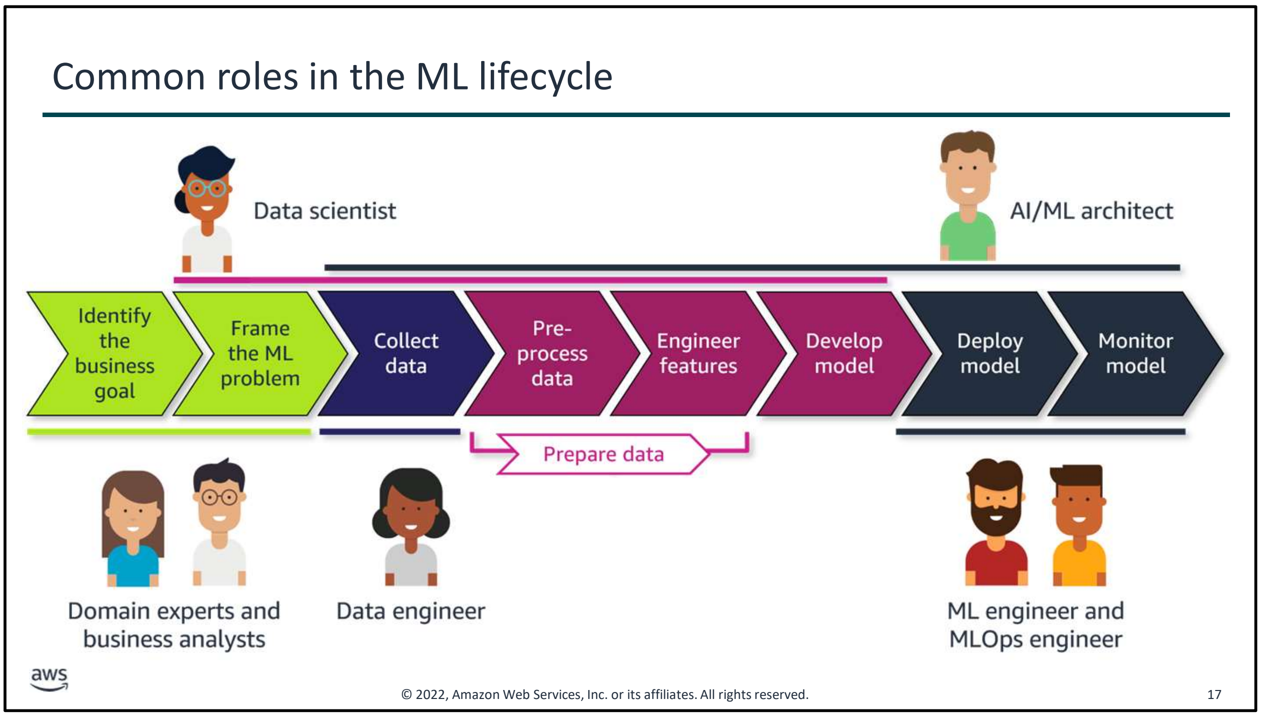


The process data phase includes collecting and preparing data



|Slide number 16
|Instructor notes
|
|Student notes

The process data phase can be segmented into two steps: collect data and prepare data. The prepare data step is split into two closely related tasks: preprocessing and feature engineering.



Common roles in the ML lifecycle

|Slide number 17
|Instructor notes
|
|Student notes

This slide identifies the types of roles that are involved in each phase of the ML lifecycle. Often, a person might play multiple roles, or the work might be split up differently across the team.

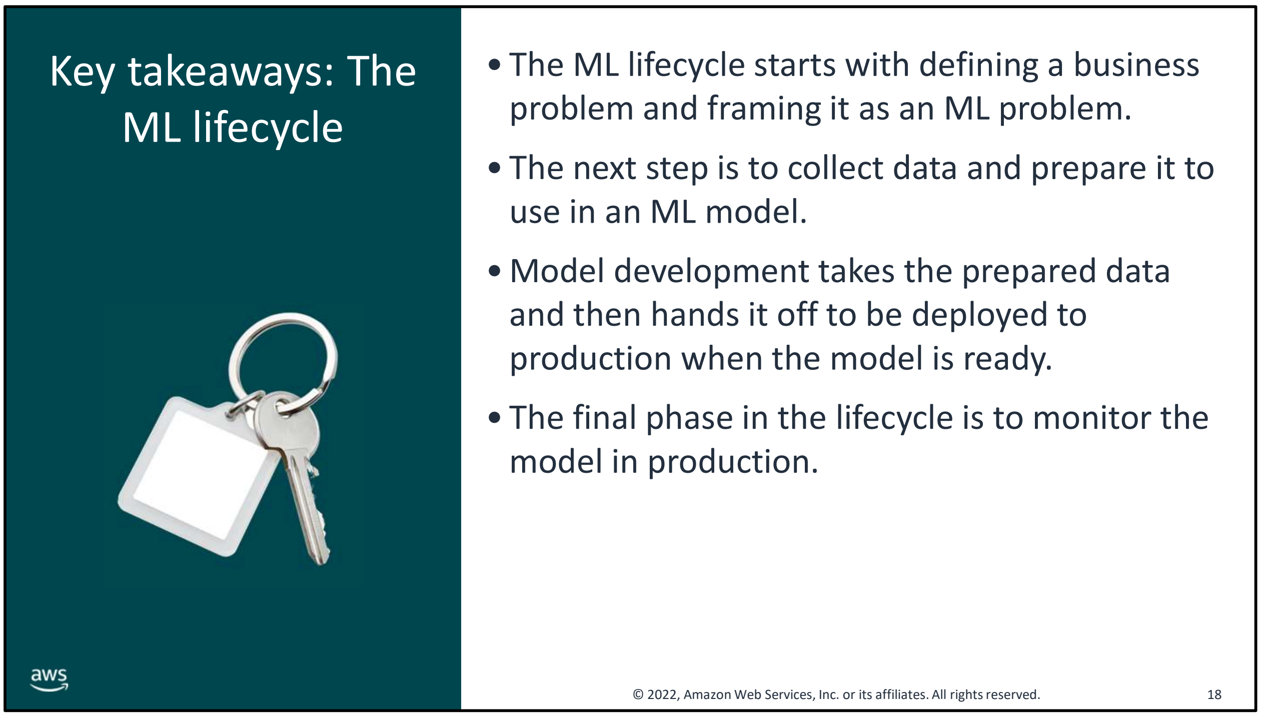
Generally speaking, a data scientist works with team members who know the relevant business domain to frame the ML problem. The data scientist is also involved in identifying the data to be collected. The data engineer is the primary owner of collecting the data and ingesting it through a data pipeline, which makes the data available for the data scientist to perform data processing on. The data scientist processes the data and develops the ML model and then hands it off to the operations team to deploy and monitor in production.

Often, an AI/ML architect will design the infrastructure and workflow components that support completion of the ML lifecycle phases.

Although a data engineer would not typically do model development or deployment, it is helpful to understand the process as it relates to the data pipeline that the engineer would build, and the amount and type of data that is needed. It's also important to recognize how feedback from the production model will come back through the pipeline.

The skills and experience of each role in the ML lifecycle overlap. In fact, data engineers and ML engineers are two of the top three roles that transition during their career into an ML data scientist role.

This module has sections about developing and deploying an ML model. The sections provide a brief overview of the tasks and environments that are required and offer additional context to data engineers about how pipeline data is used and processed in ML solutions.



Key takeaways: The ML lifecycle

- The ML lifecycle starts with defining a business problem and framing it as an ML problem.
- The next step is to collect data and prepare it to use in an ML model.
- Model development takes the prepared data and then hands it off to be deployed to production when the model is ready.
- The final phase in the lifecycle is to monitor the model in production.

|Slide number 18
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

The ML lifecycle takes a business problem and frames it as an ML problem.

The data engineer works with the data scientist to collect relevant data and prepare it for modeling through the steps of preprocessing and feature engineering.

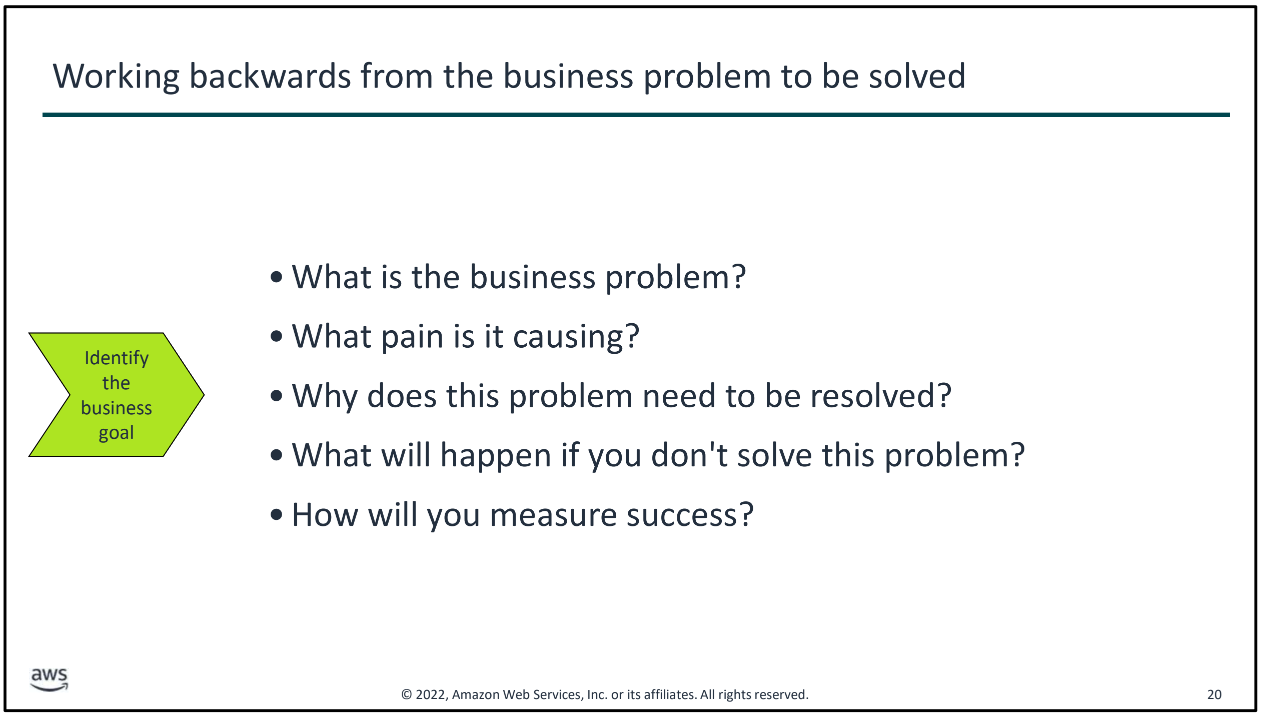
When the data scientist believes that the model is ready to be used, the operations team deploys it into production and ensures that monitoring is in place.

Most phases involve multiple roles who have interrelated specialties.

Framing the ML problem to meet the business goal

Processing Data for ML

|Slide number 19
|Instructor notes
|
|Student notes
This section describes the activities that are performed in identifying the business goal and framing the ML problem.

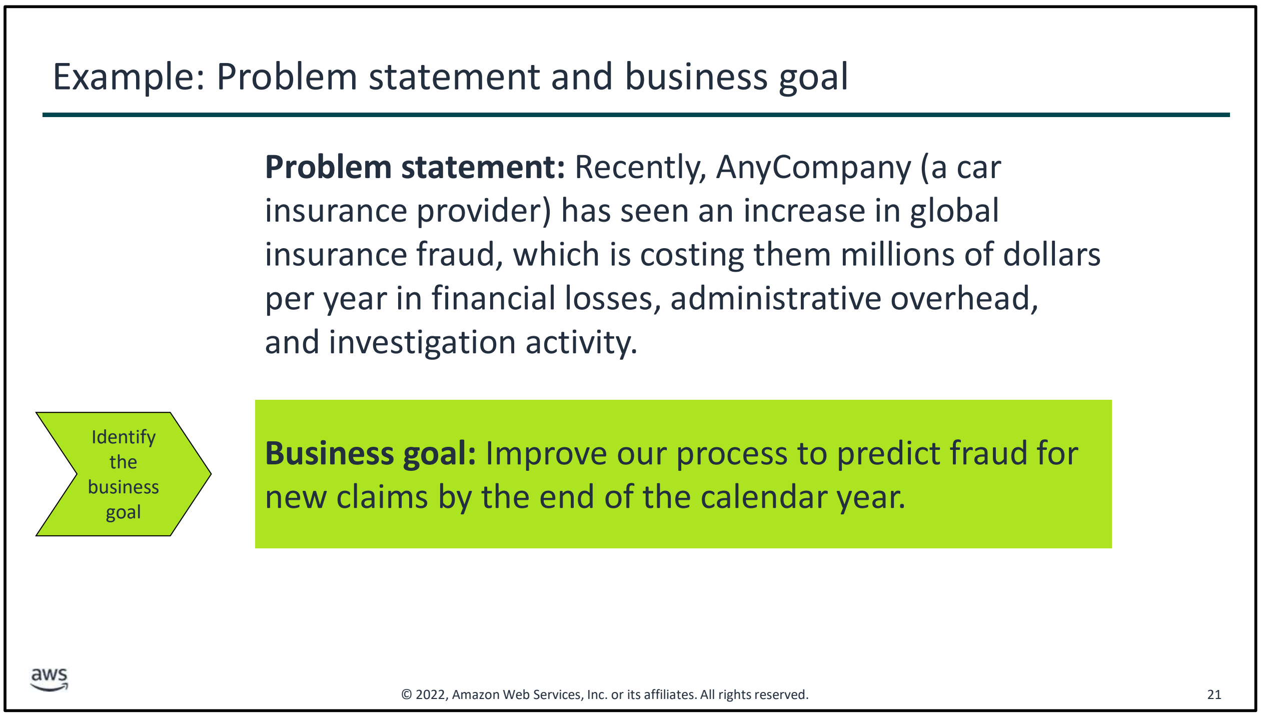


Working backwards from the business problem to be solved

Identify the business goal

- What is the business problem?
- What pain is it causing?
- Why does this problem need to be resolved?
- What will happen if you don't solve this problem?
- How will you measure success?

|Slide number 20
|Instructor notes
|
|Student notes
As described in the Data-Driven Organizations module, it is important to work backwards from the desired goal and then determine how ML might be able to address the problem. You need to ask questions that help you to clarify what you are trying to solve.



Example: Problem statement and business goal

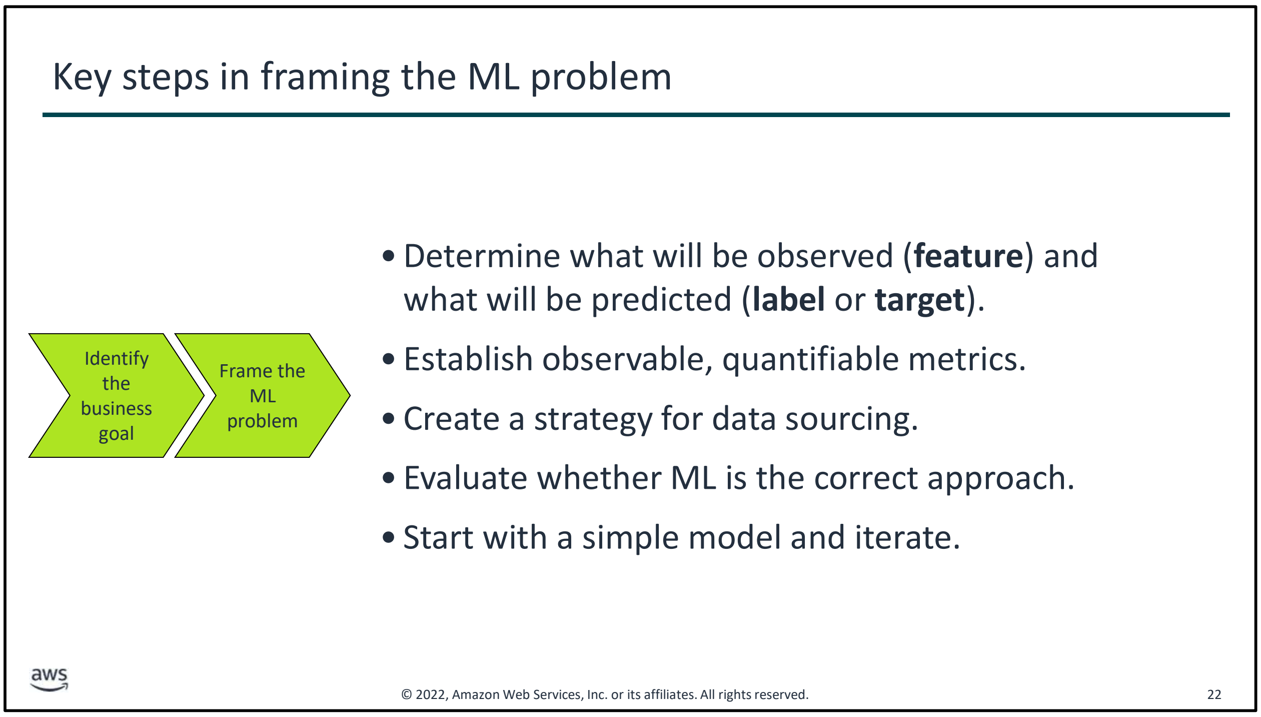
Problem statement: Recently, AnyCompany (a car insurance provider) has seen an increase in global insurance fraud, which is costing them millions of dollars per year in financial losses, administrative overhead, and investigation activity.

Identify the business goal

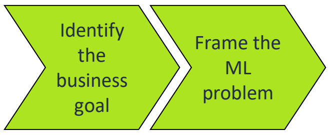
Business goal: Improve our process to predict fraud for new claims by the end of the calendar year.

|Slide number 21
|Instructor notes
|
|Student notes
The problem statement is a concise statement that summarizes the answers to the questions that were presented on the prior slide.

In the case of AnyCompany, they don't know why there is increased fraud, which types of insurance claims represent the most fraud, or where the most fraud is occurring. The ML solution could help them identify predictions so that their adjusters could address issues accordingly.



Key steps in framing the ML problem

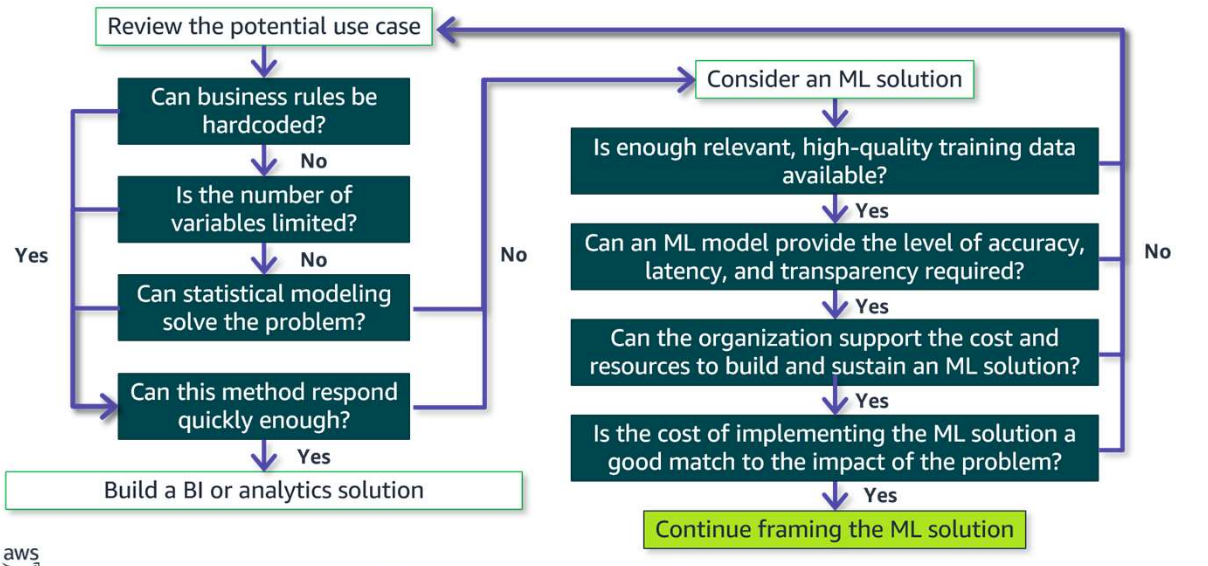


- Determine what will be observed (**feature**) and what will be predicted (**label** or **target**).
- Establish observable, quantifiable metrics.
- Create a strategy for data sourcing.
- Evaluate whether ML is the correct approach.
- Start with a simple model and iterate.

|Slide number 22
|Instructor notes
|
|Student notes

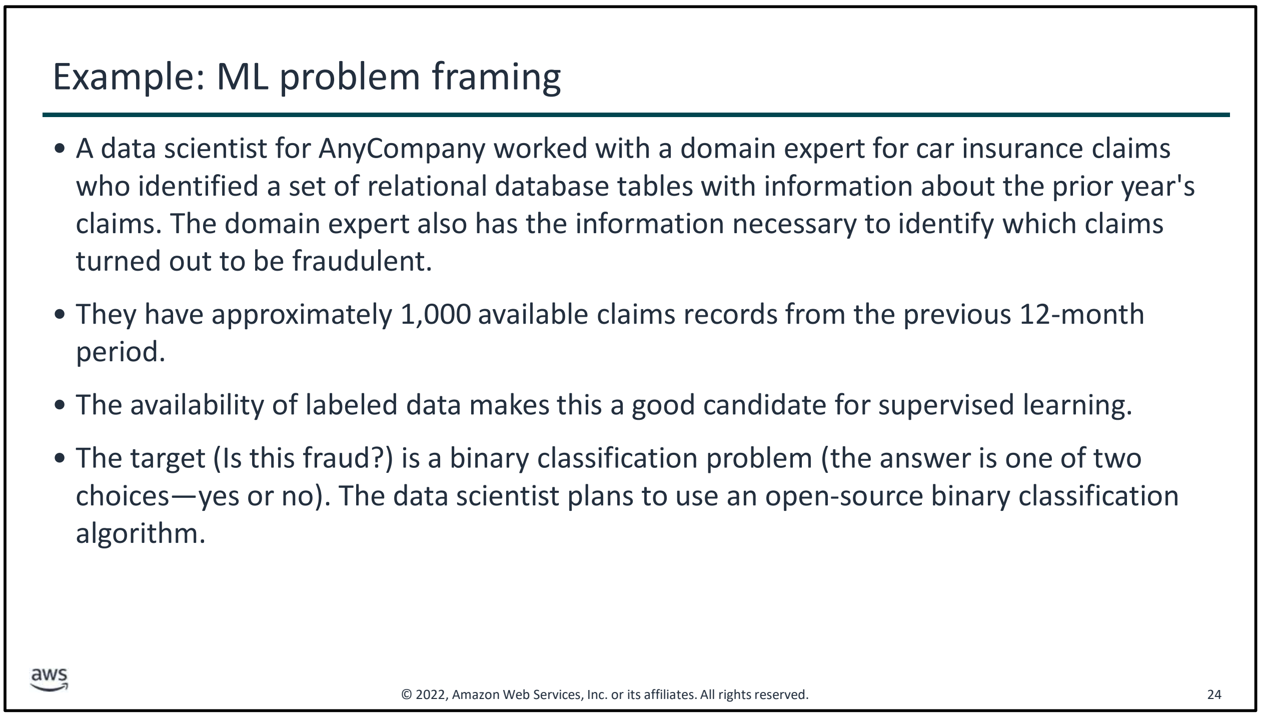
Framing the ML problem is about using domain and data science expertise to determine how (or if) ML can address the business goal. The steps listed on the slide are all part of framing the ML problem. Most importantly, you need to identify what you want to predict and where you can get the data that supports your needs.

Determining if ML is the best approach


Review the potential use case
Can business rules be hardcoded?
No
Is the number of variables limited?
No
Can statistical modeling solve the problem?
Can this method respond quickly enough?
Yes
Yes
Build a BI or analytics solution
No
Consider an ML solution
Is enough relevant, high-quality training data available?
Yes
Can an ML model provide the level of accuracy, latency, and transparency required?
Yes
Can the organization support the cost and resources to build and sustain an ML solution?
Yes
Is the cost of implementing the ML solution a good match to the impact of the problem?
Yes
Continue framing the ML solution
No

|Slide number 23
|Instructor notes
|
|Student notes
As noted on the previous slide, part of framing the ML problem is determining whether ML is the correct approach. The left side of this decision flow highlights characteristics that might point you to using a business intelligence (BI) or analytics solution instead of ML.

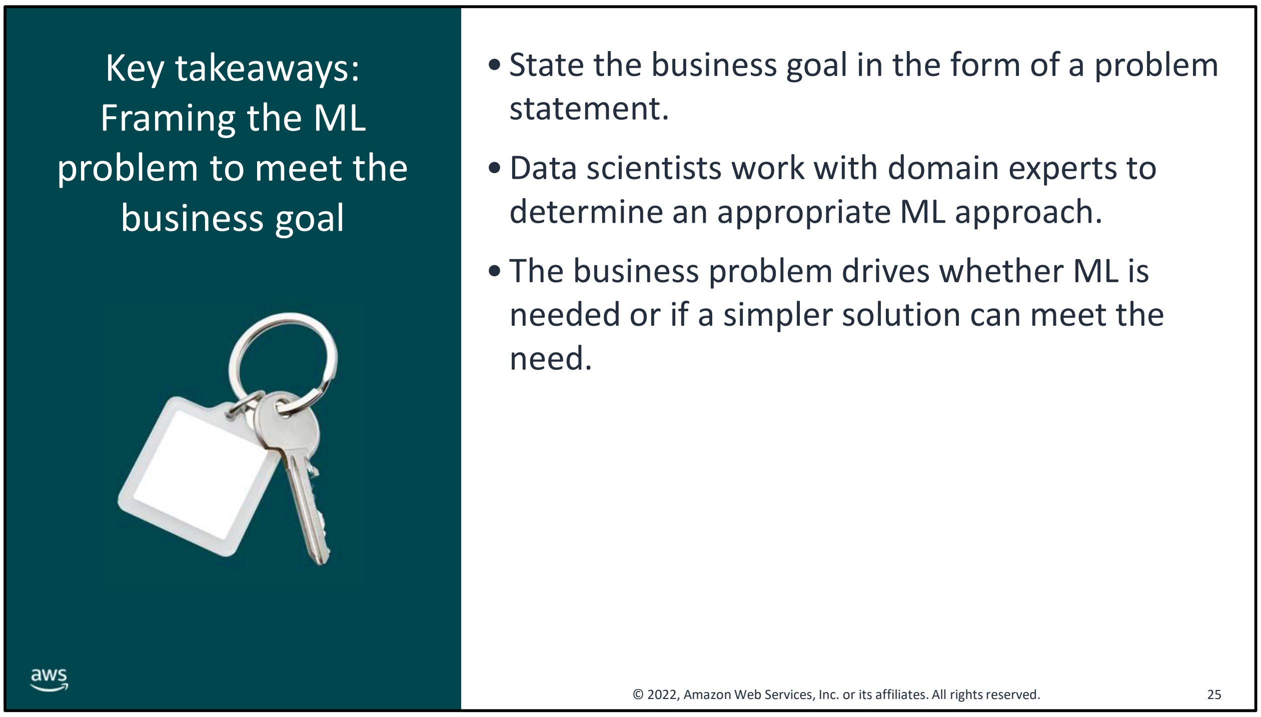
The right side of the flow highlights questions that you need to consider before embarking on an ML solution, even if the business problem seems suited to the use of ML.



Example: ML problem framing

- A data scientist for AnyCompany worked with a domain expert for car insurance claims who identified a set of relational database tables with information about the prior year's claims. The domain expert also has the information necessary to identify which claims turned out to be fraudulent.
- They have approximately 1,000 available claims records from the previous 12-month period.
- The availability of labeled data makes this a good candidate for supervised learning.
- The target (Is this fraud?) is a binary classification problem (the answer is one of two choices—yes or no). The data scientist plans to use an open-source binary classification algorithm.

|Slide number 24
|Instructor notes
|This scenario is in line with the data that is used in the SageMaker demos later in this module. The demo uses an algorithm that is available within SageMaker.
|
|Student notes
The example scenario that is described here highlights some key outcomes of the phase to frame the ML problem. The data scientist has identified a data source that has enough relevant data, and they have identified an algorithm to start from.



Key takeaways: Framing the ML problem to meet the business goal

- State the business goal in the form of a problem statement.
- Data scientists work with domain experts to determine an appropriate ML approach.
- The business problem drives whether ML is needed or if a simpler solution can meet the need.

|Slide number 25
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

Working backwards from the business goal helps to ensure that the project is focused on outcomes rather than technology.

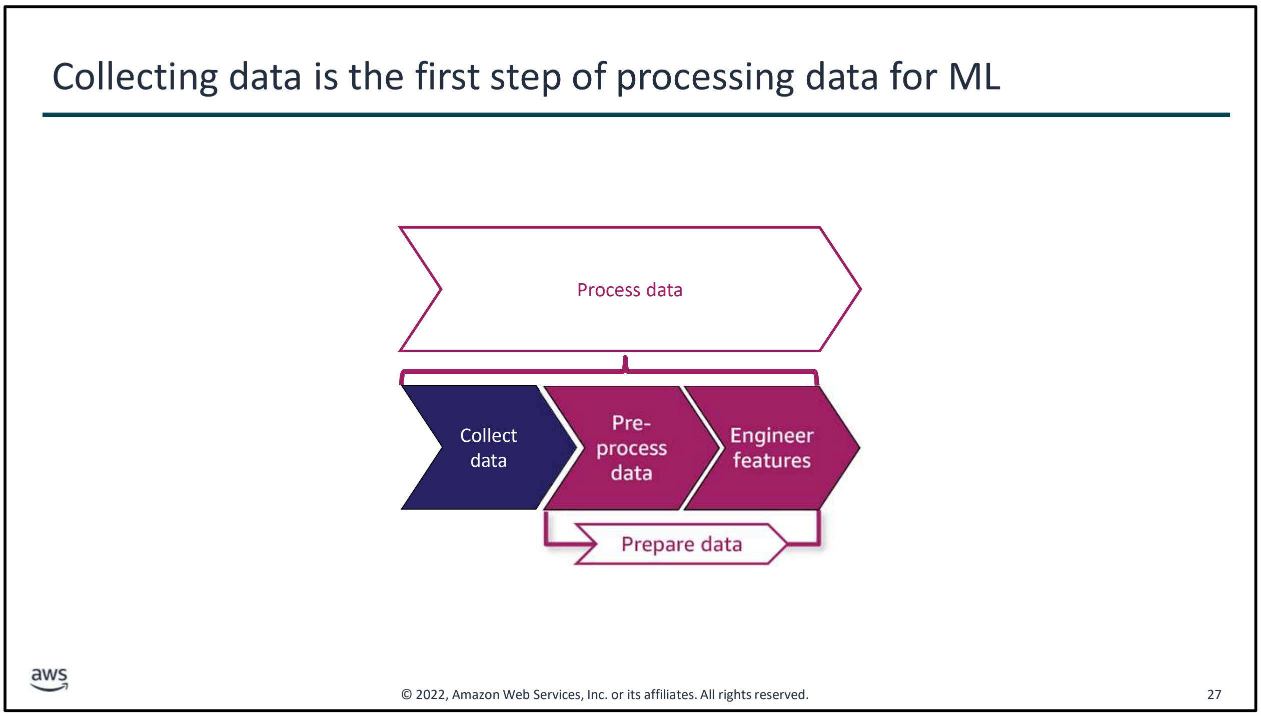
Data scientists and domain experts work together to frame the ML problem to give them a starting point to collect the appropriate data and use the correct type of algorithm.

ML is not always the correct approach. It’s important to rule out simpler approaches that might solve the requirement, rather than assuming that you need ML to predict an outcome.

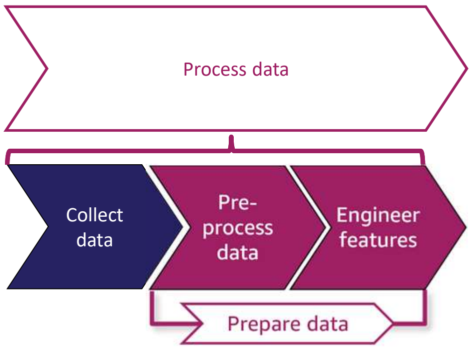
Collecting data

Processing Data for ML

|Slide number 26
|Instructor notes
|
|Student notes
This section describes the activities that are completed during the collecting data step of the processing data phase.



Collecting data is the first step of processing data for ML



~ collect step is roughly equivalent to the ingestion step we've showed. ELT to get data into the data lake. Majority of wrangling is then done by Data scientist. Cleaning is very important.

|Slide number 27

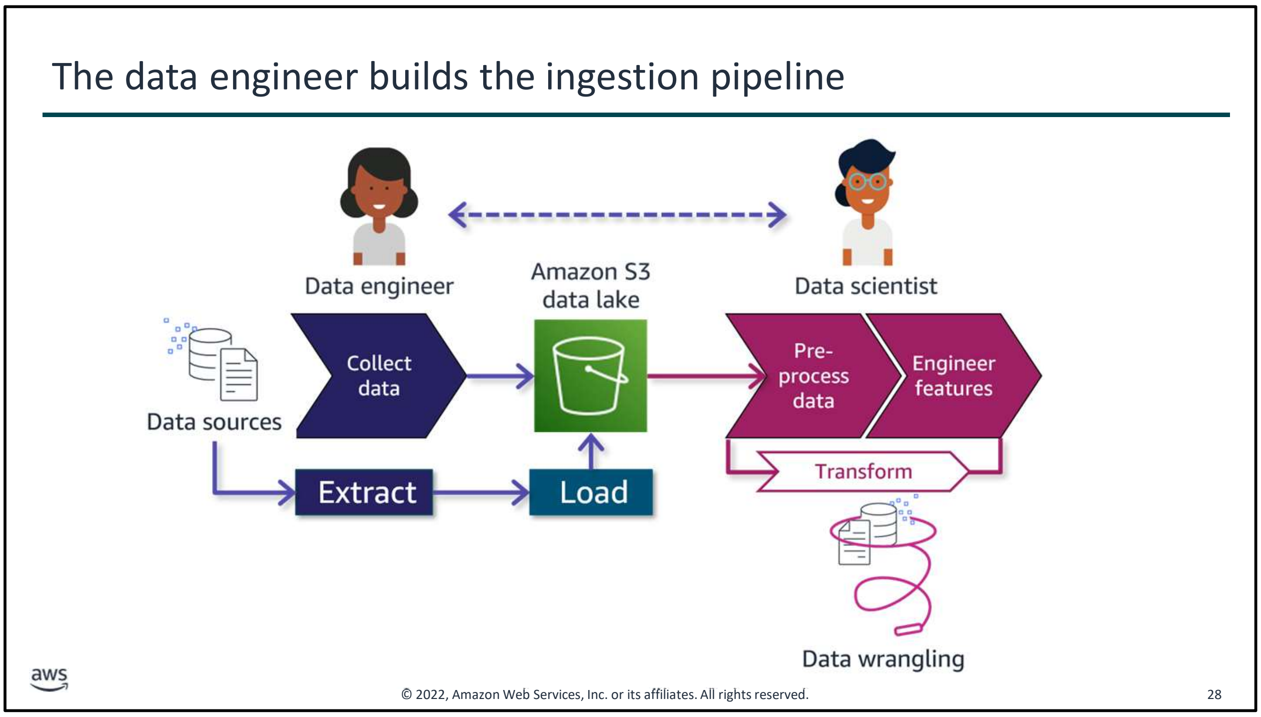
|Instructor notes

|The breakout of tasks as highlighted in the lifecycle sections of this module are based on the ML lifecycle as presented in the Machine Learning Lens of the AWS Well-Architected Framework. The notes reflect the iterative and overlapping nature of these steps, and you might find different reference materials that use different categorizations and titling for these tasks. Many of the tools that are available from AWS and other vendors to support ML model development will also blur these lines or separate them at different points. The key takeaway for students is that data should be as clean as possible before it goes to feature engineering, and the feature engineering step optimizes the available data for use with the selected algorithm.

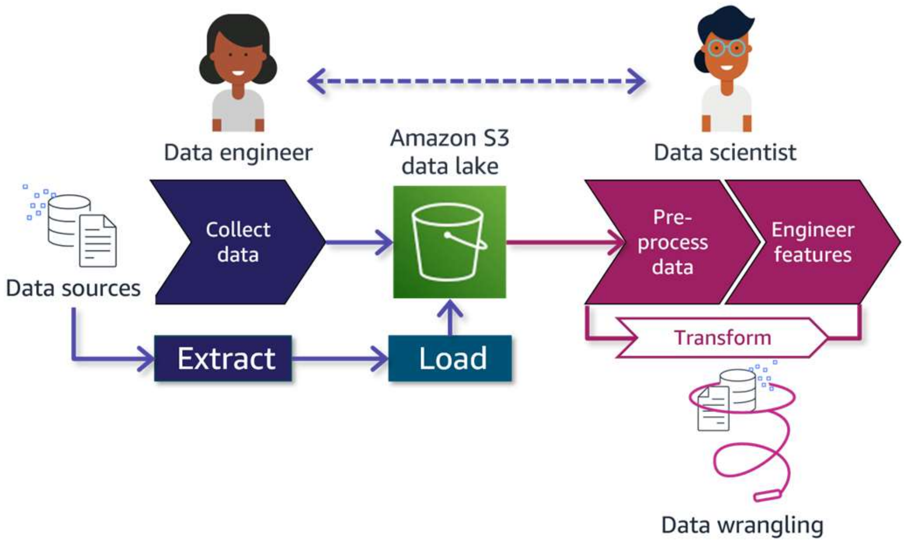
|

|Student notes

The collect data step in the ML lifecycle gets data from sources and moves it into a location where a data scientist can prepare it for ML.



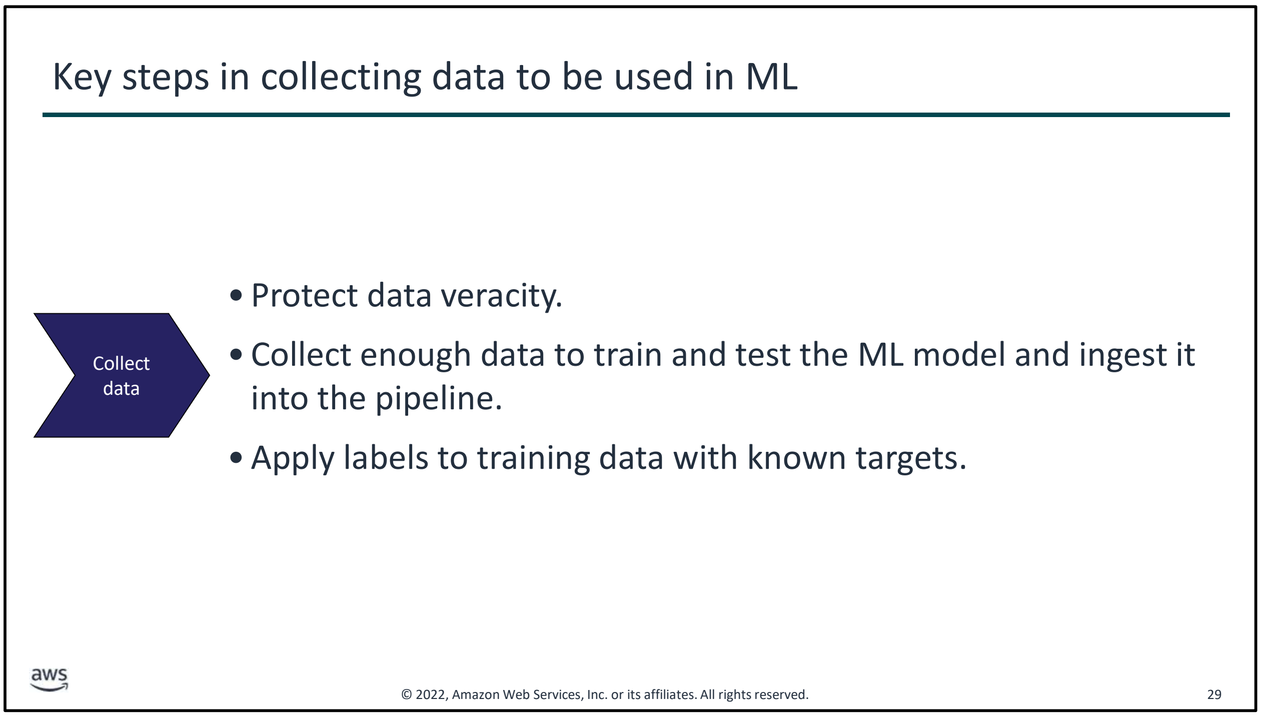
The data engineer builds the ingestion pipeline



|Slide number 28
|Instructor notes
|This slide evokes concepts that were learned in the Ingesting and Preparing Data module. An Amazon Simple Storage Service (Amazon S3) data lake is shown as the target destination for the data that is being ingested. A data lake is used as the example throughout for simplicity, although data could be loaded and made available from a variety of data stores.
|
|Student notes
Consider the ML lifecycle in the context of the data pipeline concepts that this course has presented. A common ingestion pipeline for ML would be to use an extract, load, and transform (ELT) approach to load the data into a data lake, from which most of the required transformations take place.

As noted in other modules, the steps in the process and the roles that perform them will overlap and are typically iterative. The process includes cycles of discovery, ingesting, visualizing, and processing before the ML model is built and tested. Generally, the person who performs the data engineering would be responsible for getting the data into the pipeline and loading it into the data lake. The data scientist would be responsible for the majority of the required data wrangling.

The two roles would collaborate on what data should be included and what types of structuring and cleaning are required as part of ingesting the data to support the model. The tools that are selected to perform data processing will influence where the responsibilities would be split.

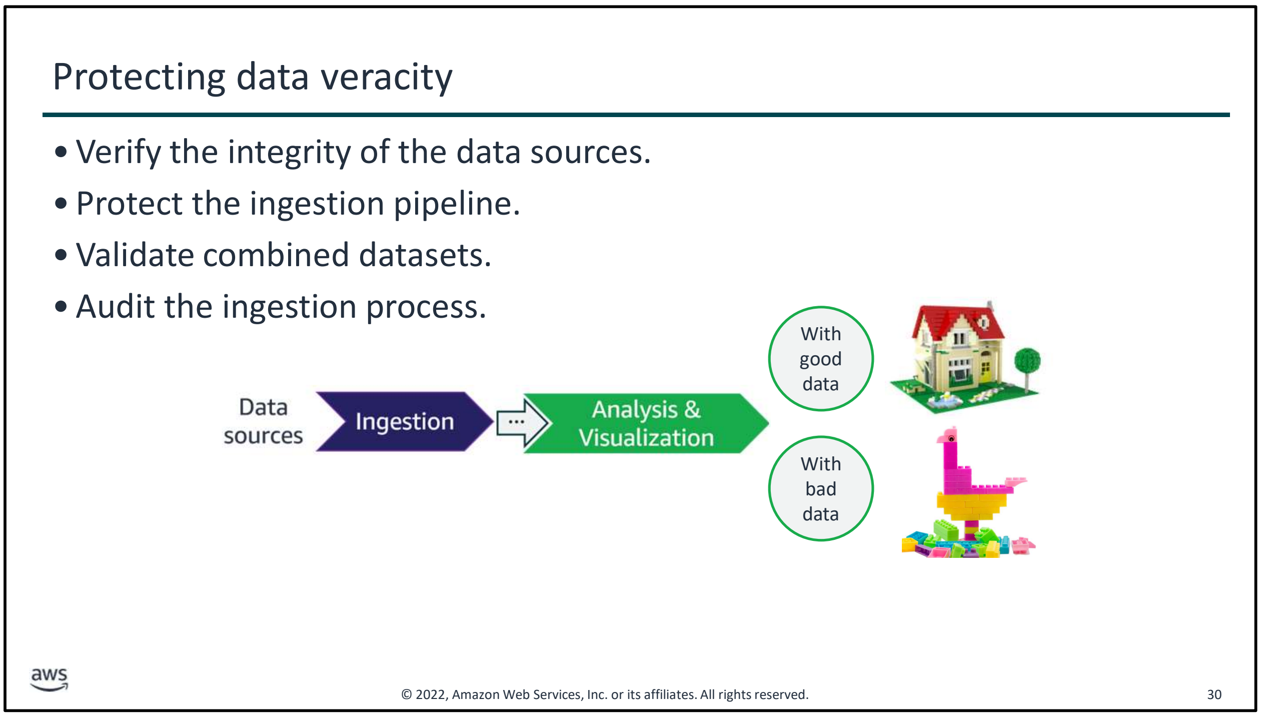


Key steps in collecting data to be used in ML

Collect data

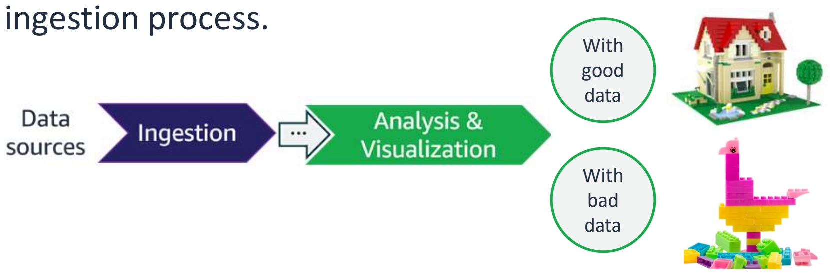
- Protect data veracity.
- Collect enough data to train and test the ML model and ingest it into the pipeline.
- Apply labels to training data with known targets.

|Slide number 29
|Instructor notes
|You might want to emphasize the fact that these are complimentary to the activities that were described in the Ingesting and Preparing Data module. These steps are not intended to describe the entire set of activities for ingesting the data.
|
|Student notes
The considerations that are described in the Ingesting and Preparing Data and the Ingesting by Batch or by Stream modules apply to ingesting data to be used in ML models. The steps that are listed here reflect specific considerations for when the data that is being ingested will be used for ML applications. The step of applying labels applies to supervised learning models. The next two slides and the section that follows describe each of these steps in more detail.



Protecting data veracity

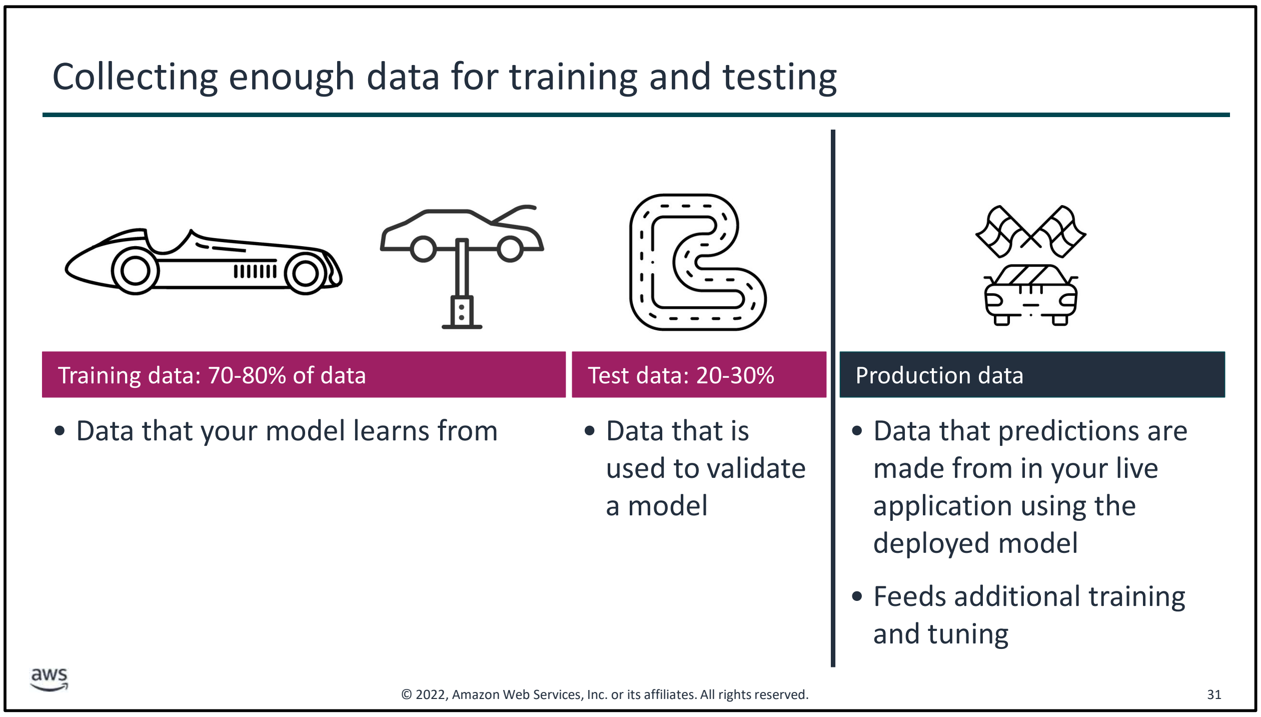
- Verify the integrity of the data sources.
- Protect the ingestion pipeline.
- Validate combined datasets.
- Audit the ingestion process.



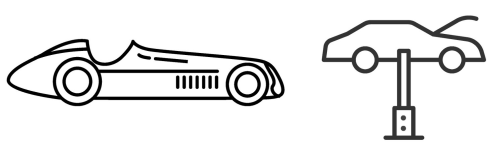
|Slide number 30
|Instructor notes
|
|Student notes
The Elements of Data module discusses the impact of bad data on analytics solutions, noting that it is better to have less high-quality data than to have a larger amount of data that cannot be trusted. Bad data is particularly harmful when fed into ML models, where it can be difficult or impossible to see the source of an error. If the model learns the wrong things, the predictions cannot be trusted. A big part of the data scientist’s job in implementing an ML solution is to make sure that the model is trained and tuned on clean data.

The data scientist might perform the majority of the cleaning tasks as part of preparing the data to work with the selected ML algorithm. However, the data engineer needs to make sure that their processes to extract data from sources and combine it to ingest into the pipeline follow best practices to ensure the veracity and secure transfer of the data that is being ingested. They should employ automated methods to validate datasets and audit the ingestion process. The data engineer might also be involved in tracing errors to their source and working with the data scientist to iterate on improving the quality of the data that is being ingested.

In the fraud detection example, the data engineer would extract data from the relational database tables that were identified during the ML problem framing phase. The engineer has good information about the reliability of the internal data source, so they can focus their efforts on applying appropriate security and monitoring features to the ingestion pipeline.



Collecting enough data for training and testing

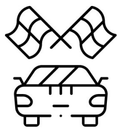


Training data: 70-80% of data

- Data that your model learns from

Test data: 20-30%

- Data that is used to validate a model



Production data

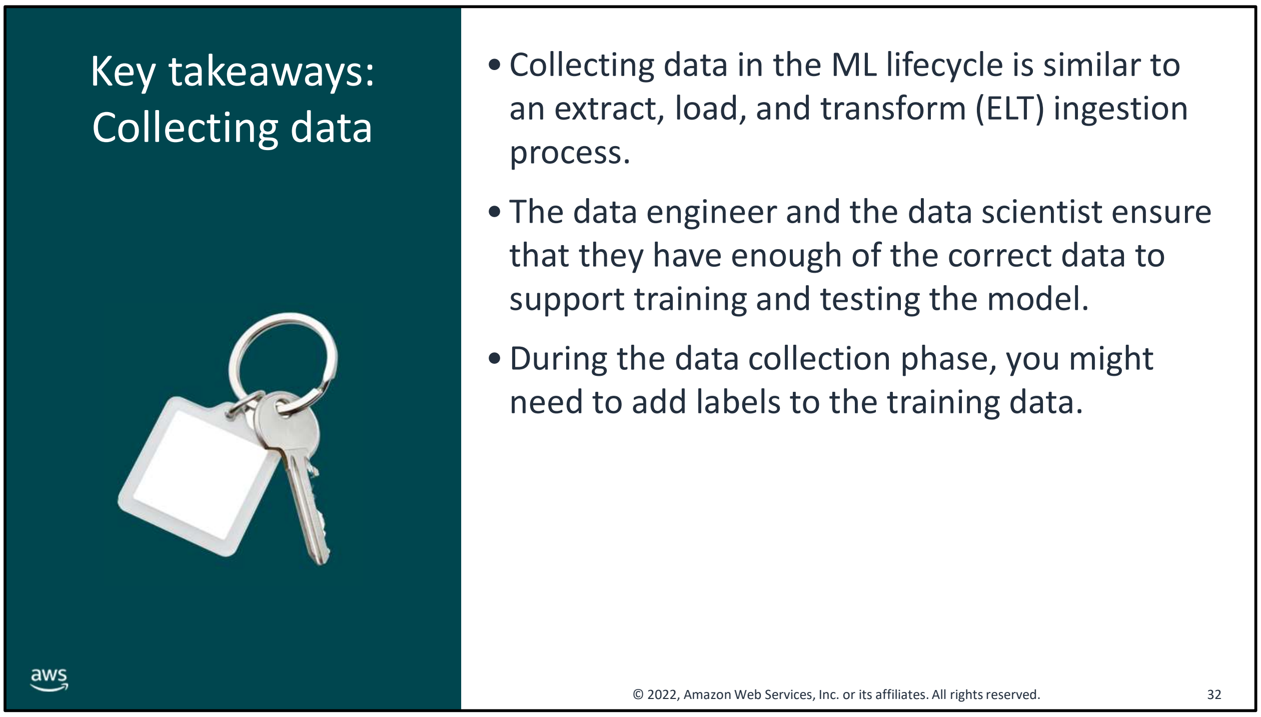
- Data that predictions are made from in your live application using the deployed model
- Feeds additional training and tuning

|Slide number 31
|Instructor notes
|
|Student notes

The dataset that is needed to build an ML model for supervised learning must provide enough quality labeled data to support both training and test datasets. The training dataset helps the model to learn the feature patterns in the data to make predictions. The test dataset helps the data scientist to validate the model and optimize the quality of the predictions. After the model has been trained and tested, it is deployed to production, where it runs against the production data source that you want to make decisions from. Feedback from production provides additional information about the accuracy of the model, and might identify issues with the model or the quality of the dataset that you used to train it.

The data engineer would not be responsible for deciding how to partition the dataset for training and test data. The data scientist would decide how to partition the data based on data characteristics and the model that is being developed. The source dataset must be large enough to account for the amount of data that needs to be labeled to train the model and the amount of test data that is needed to tune the model. The volume and quality of the available data for training and tuning are important factors in the quality of the predictions that can be made, and will vary within the type of data and modeling.

Typically, 70 to 80 percent of the collected data is used to train the model. The remaining data is used to test and tune the model. After the model is deployed to production, the actual results can also be used to train and tune the model.



Key takeaways: Collecting data

- Collecting data in the ML lifecycle is similar to an extract, load, and transform (ELT) ingestion process.
- The data engineer and the data scientist ensure that they have enough of the correct data to support training and testing the model.
- During the data collection phase, you might need to add labels to the training data.

|Slide number 32
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

The collect data step is similar to the ELT ingestion process that was described in the Ingesting and Preparing Data module.

A key aspect of this phase is to ensure that there are data sources that can provide enough high-quality data to account for training and testing the model.

Labeling data is also a task that might happen during data collection. The next section talks more about labeling.

Applying labels to training data with known targets

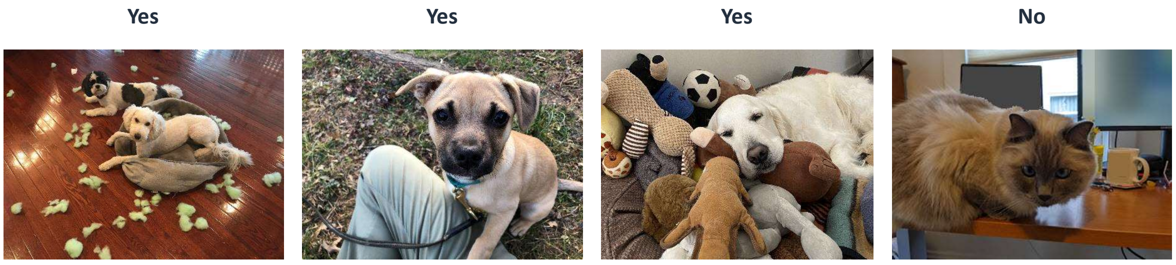
Processing Data for ML

|Slide number 33
|Instructor notes
|
|Student notes
This section describes the step of applying labels to the data that is being ingested.

Labeling provides context for the ML model

- Humans add label values to training data.
- The ML model learns from the labeled data to identify patterns.

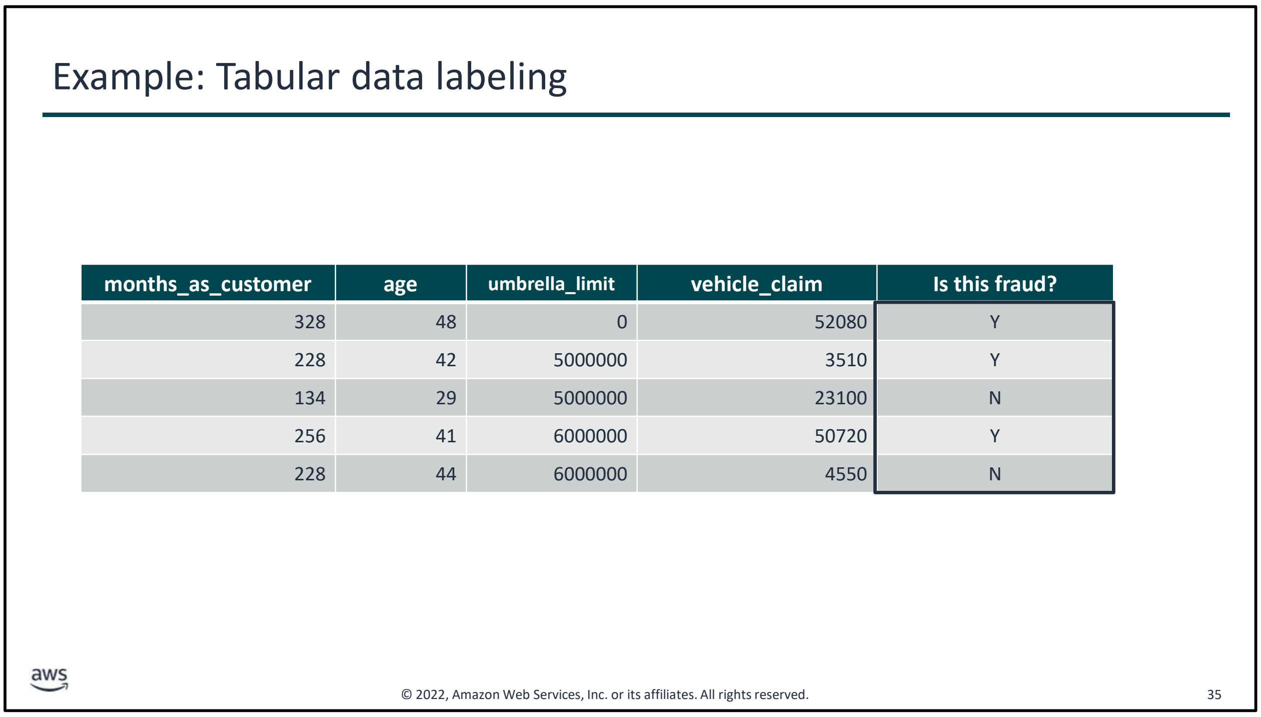
Does the photo contain a dog?



~ https://aws.amazon.com/sagemaker/data-labeling/what-is-data-labeling/
|**Slide number 34**
|**Instructor notes** Today, most machine learning tasks use a technique called supervised learning: an algorithm learns patterns or behaviors from a labeled dataset. A labeled dataset containing data samples as well as the correct answer for each one of them, also known as 'ground truth'. Depending on the problem at hand, one could use labeled images ("this is a dog", "this is a cat"), labeled text ("this is spam", "this isn't"), etc.
|
|**Student notes**
In machine learning, *data labeling* is the process of identifying targets in raw data, such as images, tabular data, text files, and videos. The process includes adding one or more meaningful and informative labels to provide context so that an ML model can learn from it. For example, a label might indicate whether a photo contains a dog or a cat, which words were uttered in an audio recording, or whether an x-ray image shows a tumor.

Data labeling typically starts by asking humans to make judgments about given pieces of unlabeled data. For example, labelers might be asked to tag all the images in a dataset where "Does the photo contain a dog?" is true. The tagging can be as rough as a simple yes or no, or as granular as identifying the specific pixels in the image that are associated with the dog. The ML model uses human-provided labels to learn the underlying patterns in a process that is called model training. The result is a trained model, which can be used to make predictions on new data.

A properly labeled dataset is used as the objective standard to train and assess a model, so spending the time and resources to ensure highly accurate data labeling is essential.

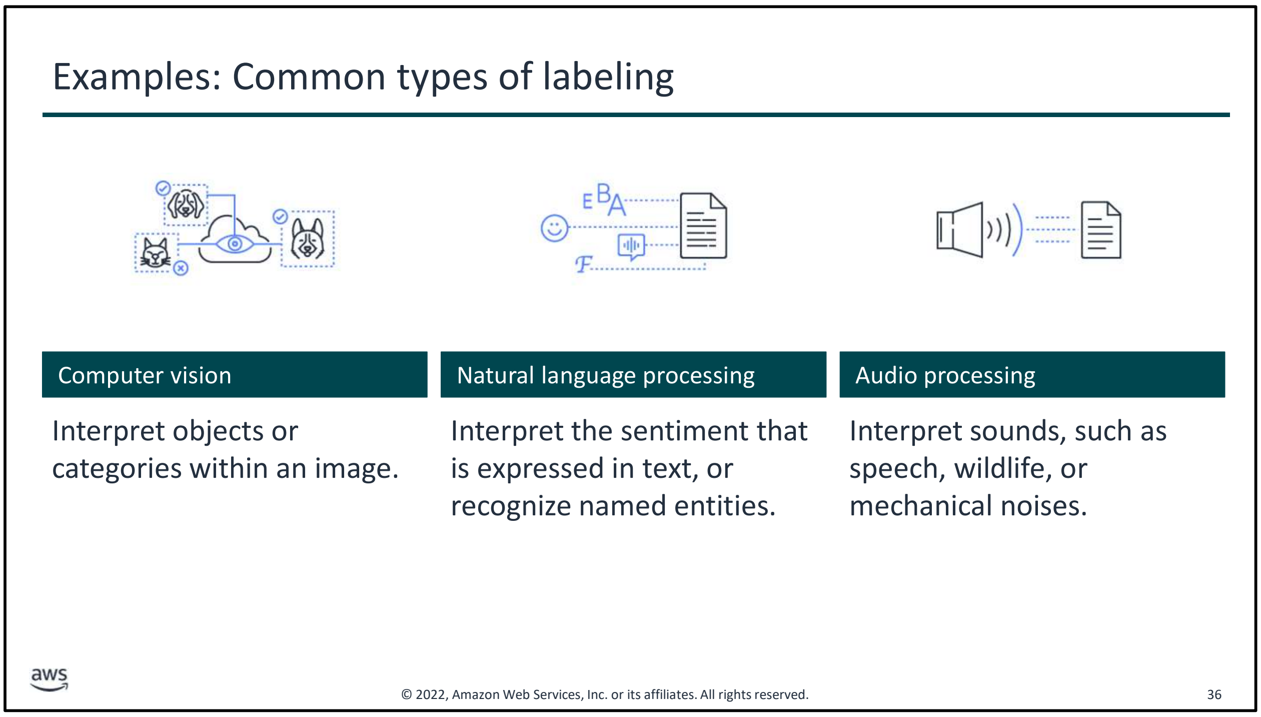


Example: Tabular data labeling

months_as_customer	age	umbrella_limit	vehicle_claim	Is this fraud?
328	48	0	52080	Y
228	42	5000000	3510	Y
134	29	5000000	23100	N
256	41	6000000	50720	Y
228	44	6000000	4550	N

|Slide number 35
|Instructor notes
|
|Student notes

In the fraud example, the labeling task would be to identify which samples in the ingested dataset were, in fact, fraudulent. The "Is this fraud?" column is highlighted showing that this column is where you would capture the label (or target). This would enable the model to find patterns in the features that are related to that label and then predict which other samples are fraudulent.



Examples: Common types of labeling

Computer vision	Natural language processing	Audio processing
Interpret objects or categories within an image.	Interpret the sentiment that is expressed in text, or recognize named entities.	Interpret sounds, such as speech, wildlife, or mechanical noises.

~ images from what is labeling page on https://aws.amazon.com/sagemaker/data-labeling/what-is-data-labeling/

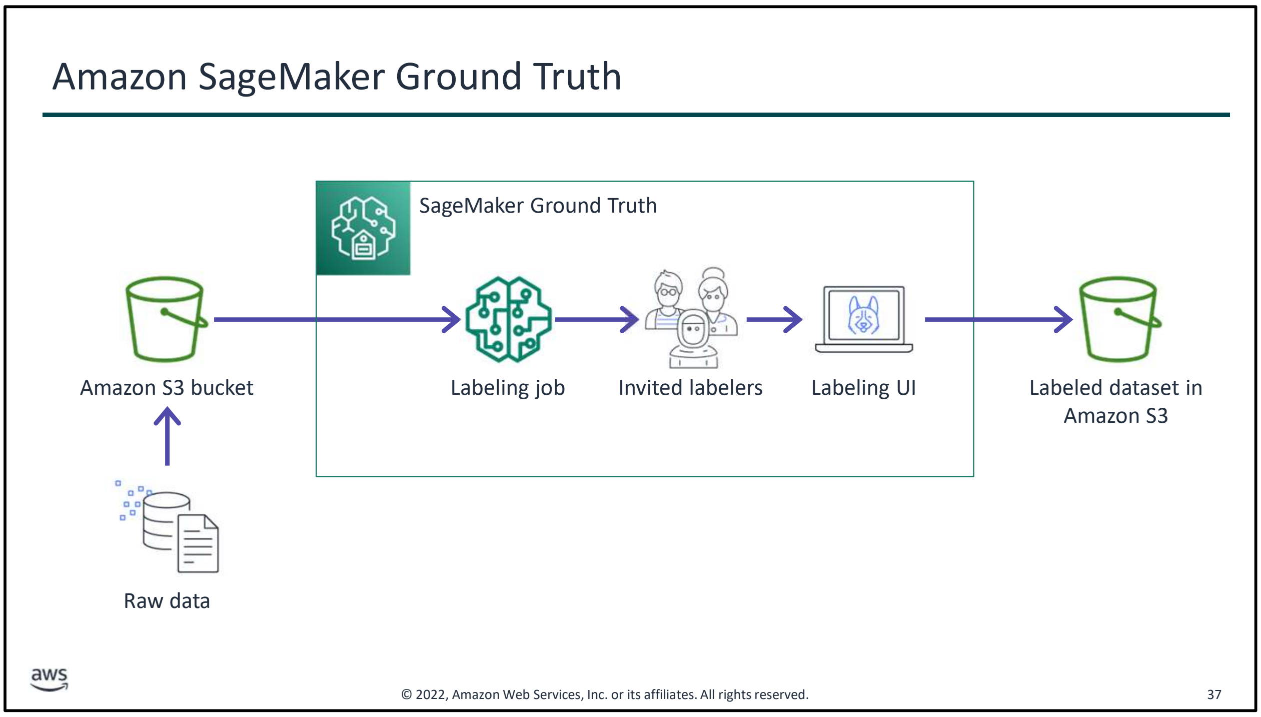
|Slide number 36

|Instructor notes

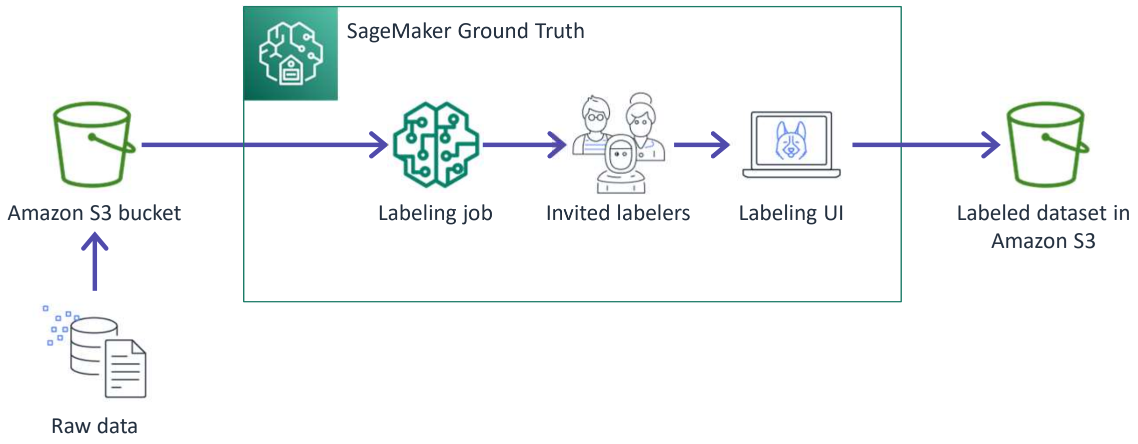
|

|Student notes

As noted in the slide, labeling might be done on images, text, or sounds.



Amazon SageMaker Ground Truth



~ provide instructors with guidance on how to do a labeling job in learner lab, or possibly quick demo. Dave Mohr checking on feasibility.
~redraw diagram from docs
|Slide number 37
|Instructor notes This diagram shows the flow of data into Amazon SageMaker where a labeling job can be created with SageMaker Ground Truth. The activity that follows shows this flow in action.
|
|Student notes
Amazon SageMaker Ground Truth is an example of a tool that makes it easier to organize and complete labeling tasks. You can upload a set of unlabeled data to an Amazon Simple Storage Service (Amazon S3) bucket, then set up a labeling job in SageMaker Ground Truth. In the labeling job, you can invite participants to be your labeling workforce and provide instructions and a UI that makes it easy for the workforce to apply labels to each element in the job.

Those who are invited to work on the labeling job receive a URL in an email invitation that takes them to the UI. Here they can label items that are stored in the S3 bucket that is associated with the job. The results from the labeling job are stored in Amazon S3, ready for additional preparation.

Activity: Labeling with SageMaker Ground Truth

- Your customer has a pet supply website where pet owners can upload photos of their pets. The customer wants the website to automatically categorize new uploads as dog, cat, or other.
- You and your classmates are the labeling workforce.

|Slide number 38

|Instructor notes

|Instructions for this activity are included in the instructor guide. You can start the labeling job in Amazon SageMaker Ground Truth and invite students by email to label images. They will get an email invitation and will need to sign in and update the temporary password to be able to perform the labeling activity. The file with the images for the activity is included in the instructor files section of the course.

|

|Student notes

Use the guidance that your instructor provides to take part in labeling the existing photos as training data by using SageMaker Ground Truth.

Key takeaways: Applying labels to training data with known targets

- Labeling is the process of assigning useful labels to targets in the training data.
- Common types of labeling include computer vision, natural language processing, audio processing, and tabular data.
- By using tools such as SageMaker Ground Truth, you can share labeling jobs with an expert workforce.

|Slide number 39
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

Labeling is the act of adding useful labels to targets in the training data.

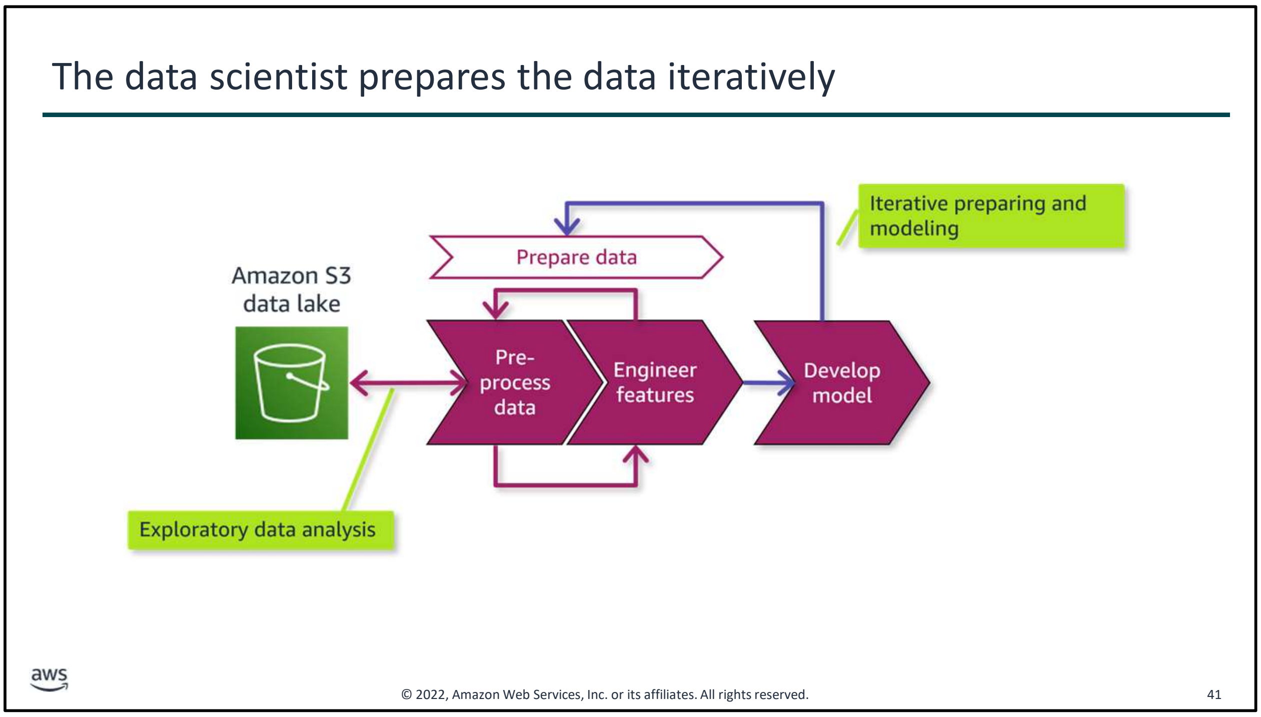
Labeling can be done on images and video for computer vision, on text for natural language processing, and on audio files to categorize sounds.

Tools, including SageMaker Ground Truth, can simplify the ability to give experts access to a dataset and apply labels to it.

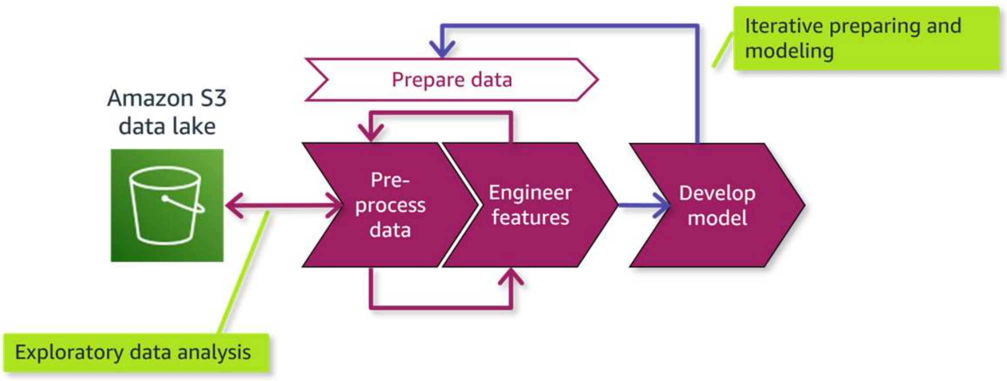
Preprocessing data

Processing Data for ML

|Slide number 40
|Instructor notes
|
|Student notes
This section describes the preprocess data task within the prepare data step of the ML lifecycle.



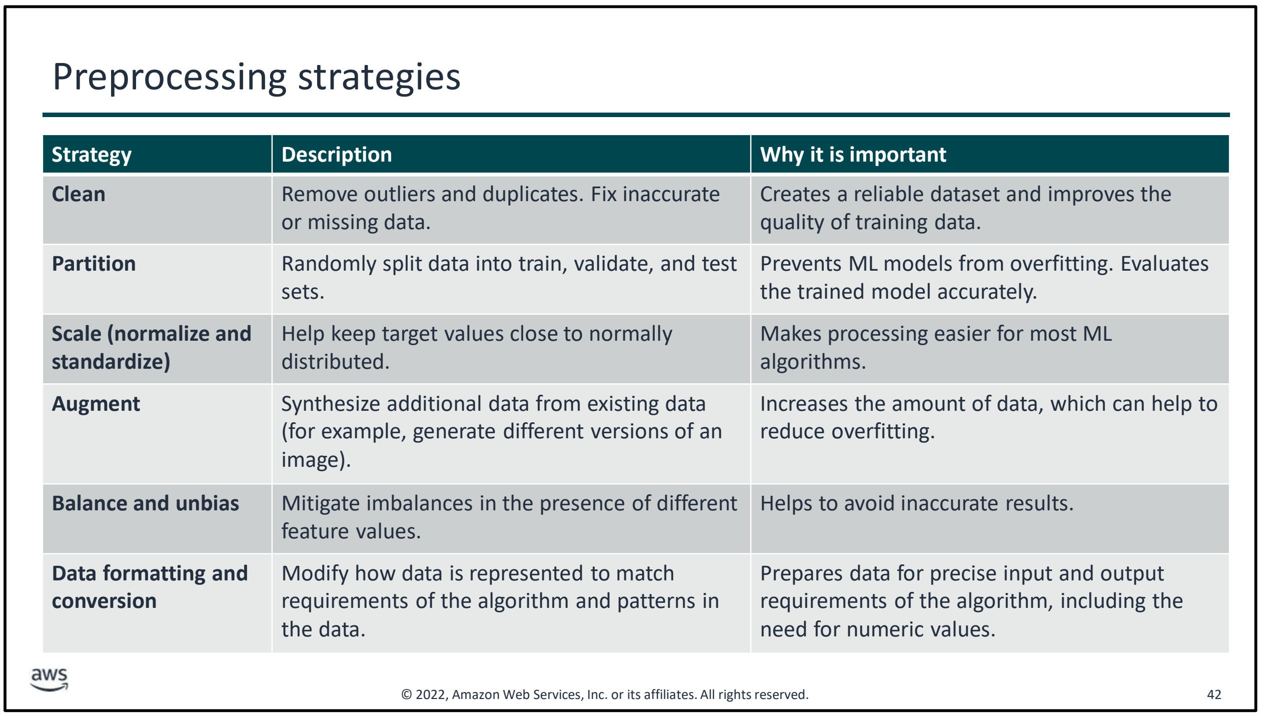
The data scientist prepares the data iteratively



~reference to Jupyter notebooks, pandas, PySpark – introduced in big data?
|Slide number 41
|Instructor notes
|
|Student notes
As illustrated on the slide, the steps of data preprocessing, feature engineering, and model development are all iterative and intertwined. Data preprocessing puts data into the correct shape and quality for training. With ML data processing, it's mostly the data scientist's role to prepare and wrangle the data. Their activities are similar to the data wrangling steps that were presented in the Ingesting and Preparing Data module, but ML preprocessing includes activities that are specific to preparing data for ML models. The data scientist might also request updates to the transformations that are included in the data collection process, if their exploration of the data suggests that changes would benefit the processing.

An important component of the data preparation step is the analysis and visualization of the data as it's being explored and processed. Exploratory data analysis (EDA) with visualization tools can help a data scientist to quickly gain a deeper understanding of data.

This helps to identify patterns that are not evident by looking at data in tables. Data scientists might employ a variety of tools to serve this purpose, some of which let them combine data exploration, wrangling, and interactive analysis and visualization in an integrated way.



Preprocessing strategies

Strategy	Description	Why it is important
Clean	Remove outliers and duplicates. Fix inaccurate or missing data.	Creates a reliable dataset and improves the quality of training data.
Partition	Randomly split data into train, validate, and test sets.	Prevents ML models from overfitting. Evaluates the trained model accurately.
Scale (normalize and standardize)	Help keep target values close to normally distributed.	Makes processing easier for most ML algorithms.
Augment	Synthesize additional data from existing data (for example, generate different versions of an image).	Increases the amount of data, which can help to reduce overfitting.
Balance and unbias	Mitigate imbalances in the presence of different feature values.	Helps to avoid inaccurate results.
Data formatting and conversion	Modify how data is represented to match requirements of the algorithm and patterns in the data.	Prepares data for precise input and output requirements of the algorithm, including the need for numeric values.

~ Overfitting occurs when you achieve a good fit of your model on the training data, while it does not generalize well on new, unseen data. In other words, the model learned patterns specific to the training data, which are irrelevant in other data.
~ https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/data-preprocessing.html

|Slide number 42

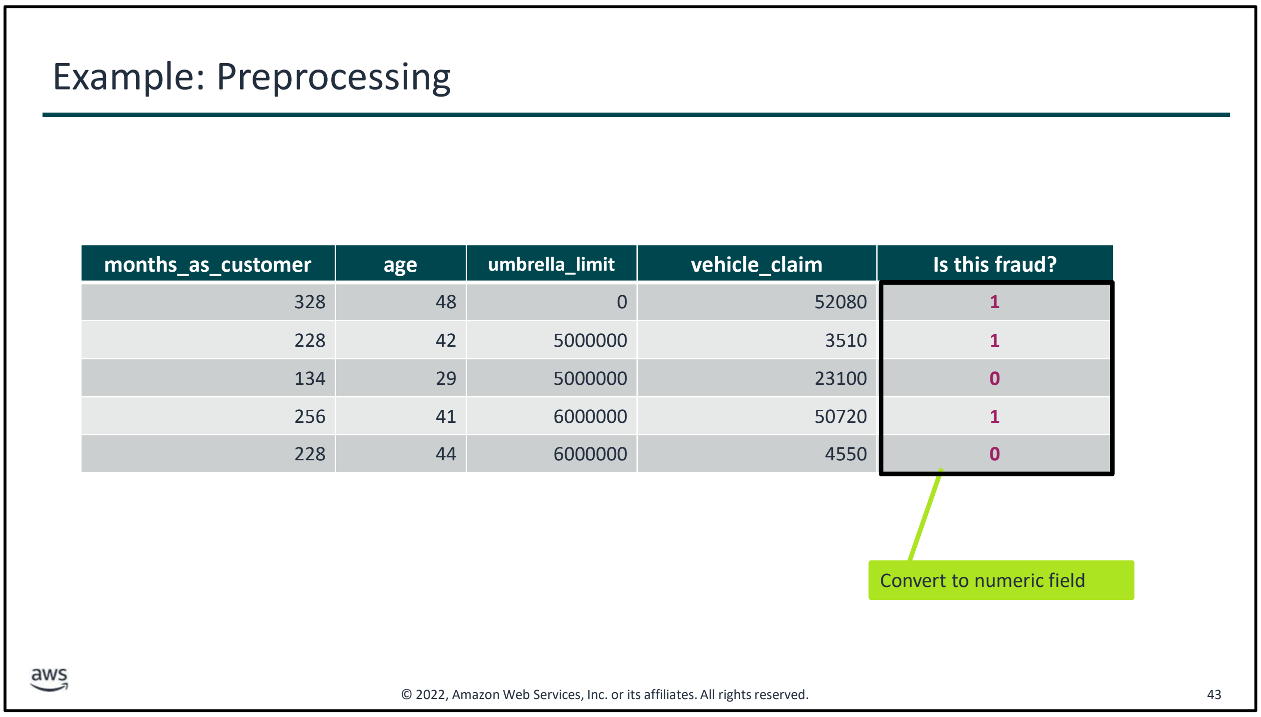
|Instructor notes

|

|Student notes

Many data preprocessing strategies exist, including those listed here. Data scientists would apply the appropriate strategies based on the dataset and the intended use case. This might include a variety of statistical modeling techniques or steps, such as creating rotated or resized versions of an image to augment an existing set of images. Many of the individual strategies that are listed actually contribute to the important strategy of balancing—or unbiasing—data. To produce accurate predictions, it’s important that the data scientist identify and mitigate bias in the data. For example, if the goal is to predict outcomes for a global customer base, but the dataset includes a large percentage of samples from one country, the results would be biased.

Data wrangling tools can simplify application of these strategies. Preprocessing is closely tied to feature engineering, and you might find overlap in how different ML tools or references categorize preprocessing activities compared to feature engineering activities. The goal of preparing data is to provide a clean, balanced dataset that the desired algorithm can use and will produce accurate results. Preprocessing focuses on getting the dataset to where the data scientist can perform the more advanced feature engineering tasks.



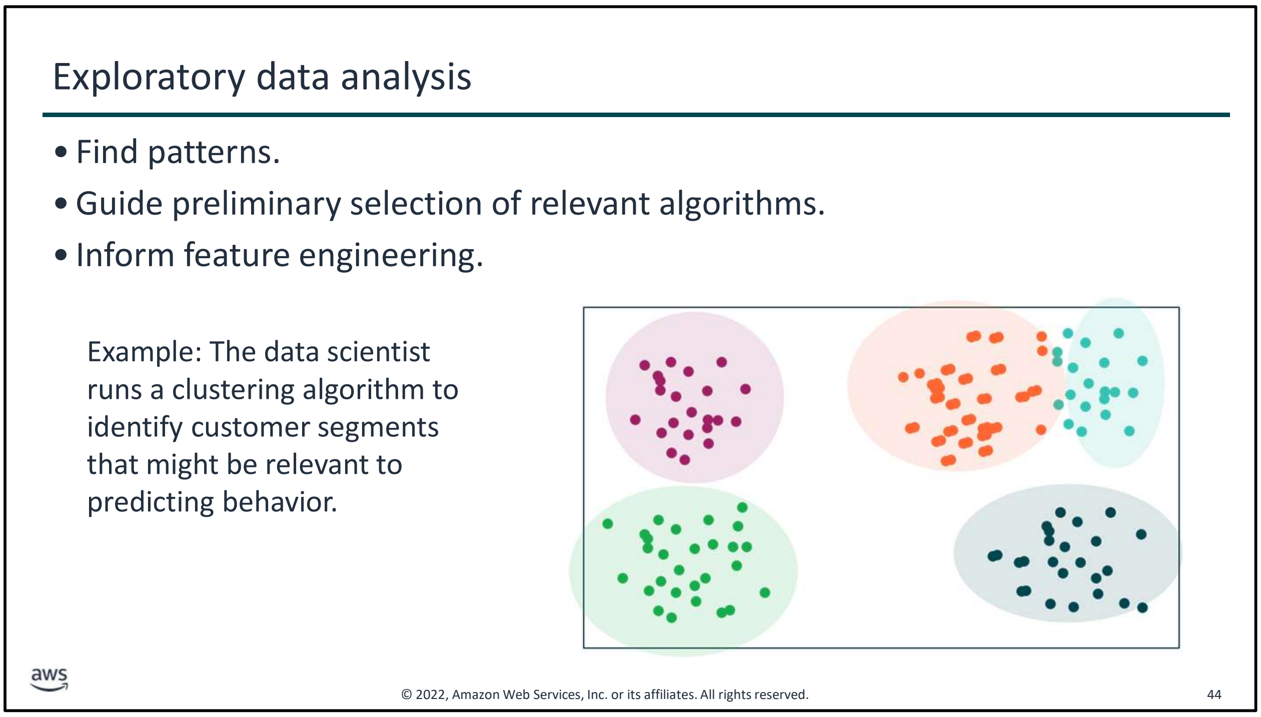
Example: Preprocessing

months_as_customer	age	umbrella_limit	vehicle_claim	Is this fraud?
328	48	0	52080	1
228	42	5000000	3510	1
134	29	5000000	23100	0
256	41	6000000	50720	1
228	44	6000000	4550	0

Convert to numeric field

|Slide number 43
|Instructor notes
|
|Student notes

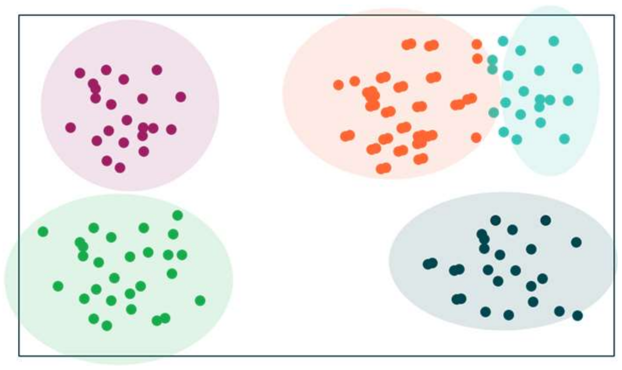
A preprocessing task for the fraud detection example might be to convert the Y or N in the "Is this fraud?" field to a numeric equivalent, such as 1 for yes and 0 for no. This task can simplify the algorithm's ability to work with the field.



Exploratory data analysis

- Find patterns.
- Guide preliminary selection of relevant algorithms.
- Inform feature engineering.

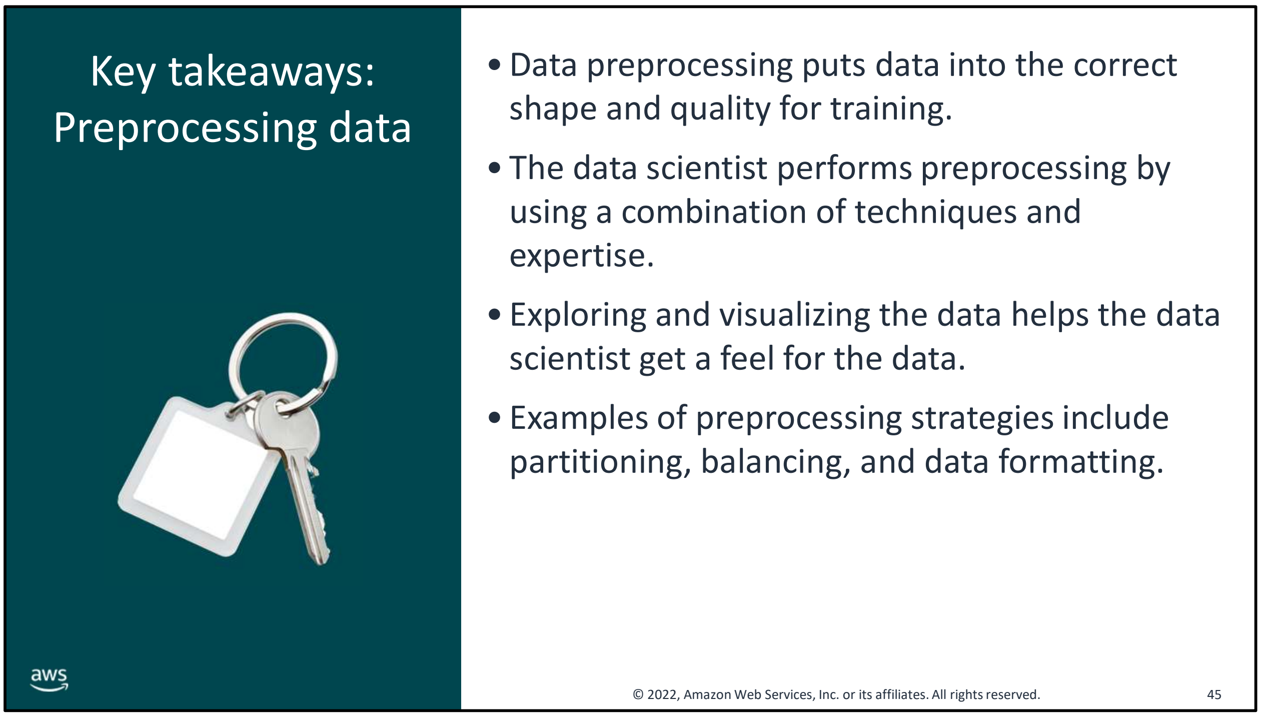
Example: The data scientist runs a clustering algorithm to identify customer segments that might be relevant to predicting behavior.



|Slide number 44
|Instructor notes
|
|Student notes

Data exploration typically begins with desktop tools, including Excel and database access tools. But often, data scientists will use interactive Jupyter notebooks with tools such as pandas and PySpark to develop visualizations of the data. Pandas is an open-source data analysis and manipulation tool that is built on top of the Python programming language. PySpark is the Python API for Apache Spark, and it provides a shell to interactively analyze data in an Apache Spark environment.

This type of analysis provides the data scientist with insights to help them decide what is important and can provide insight into feature engineering. For example, a data scientist might use a clustering algorithm to identify categories within the dataset, such as customer segments. Clustering algorithms are an important part of unsupervised learning models but can also be applied to data exploration. The clustering can inform how features are prioritized or what type of additional feature engineering might be needed to make the dataset more useful.



Key takeaways: Preprocessing data

- Data preprocessing puts data into the correct shape and quality for training.
- The data scientist performs preprocessing by using a combination of techniques and expertise.
- Exploring and visualizing the data helps the data scientist get a feel for the data.
- Examples of preprocessing strategies include partitioning, balancing, and data formatting.

|Slide number 45
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

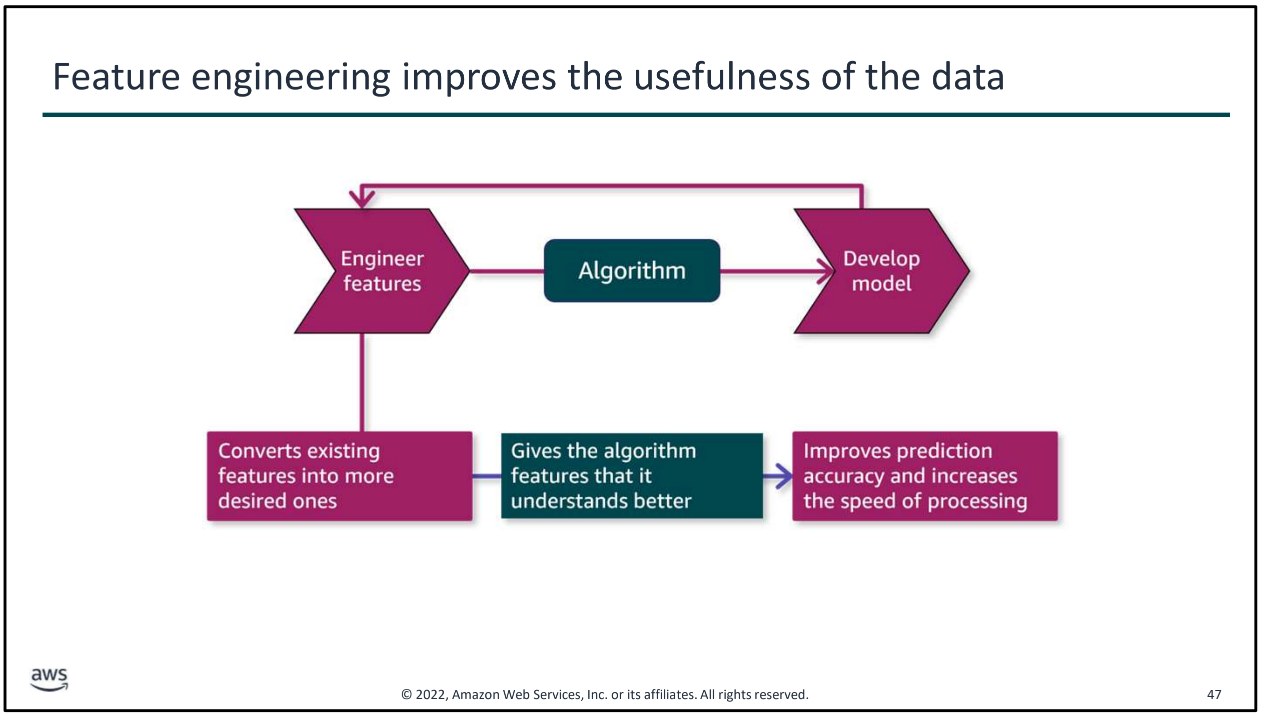
The goal of data preprocessing is to start preparing the data for training.

The data scientist usually does preprocessing, and they would usually explore the data and perform visualizations to help them identify what needs to be changed. They might use a variety of strategies during preprocessing.

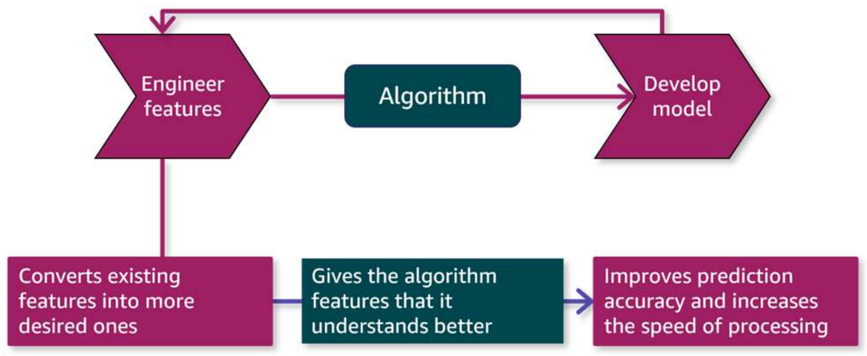
Feature engineering

Processing Data for ML

|Slide number 46
|Instructor notes
|
|Student notes
This section describes the feature engineering task within the prepare data step of the ML lifecycle.



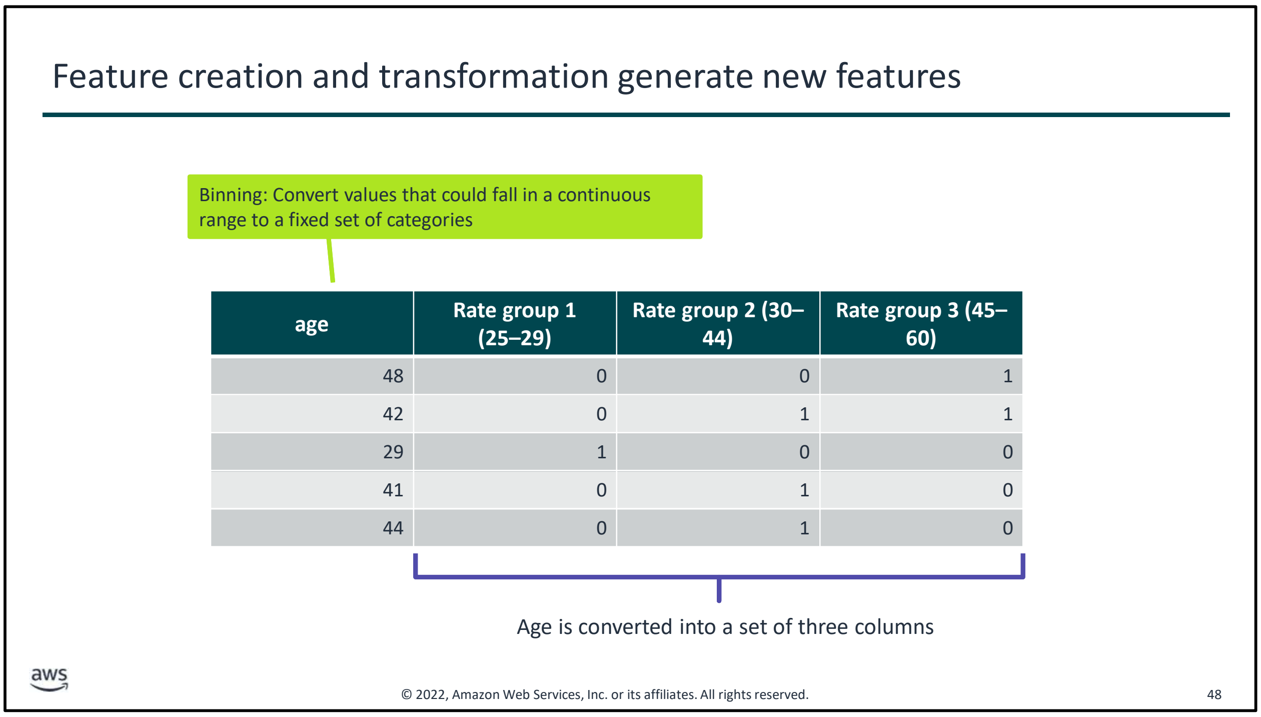
Feature engineering improves the usefulness of the data


Engineer features
Algorithm
Develop model
Converts existing features into more desired ones
Gives the algorithm features that it understands better
Improves prediction accuracy and increases the speed of processing

|Slide number 47
|Instructor notes
|
|Student notes

Feature engineering is a process to select and transform variables (features) in the dataset to improve how the selected algorithm can use them. Feature engineering is also used to improve the accuracy and performance of the model. Data scientists apply domain knowledge and statistical modeling to perform feature engineering. In some cases, the feature engineering step itself can use machine learning to perform feature engineering!

The tasks of feature engineering include feature creation, transformation, extraction, and selection. The next two slides provide examples for these steps.



Feature creation and transformation generate new features

Binning: Convert values that could fall in a continuous range to a fixed set of categories

age	Rate group 1 (25–29)	Rate group 2 (30–44)	Rate group 3 (45–60)
48	0	0	1
42	0	1	1
29	1	0	0
41	0	1	0
44	0	1	0

Age is converted into a set of three columns

|Slide number 48

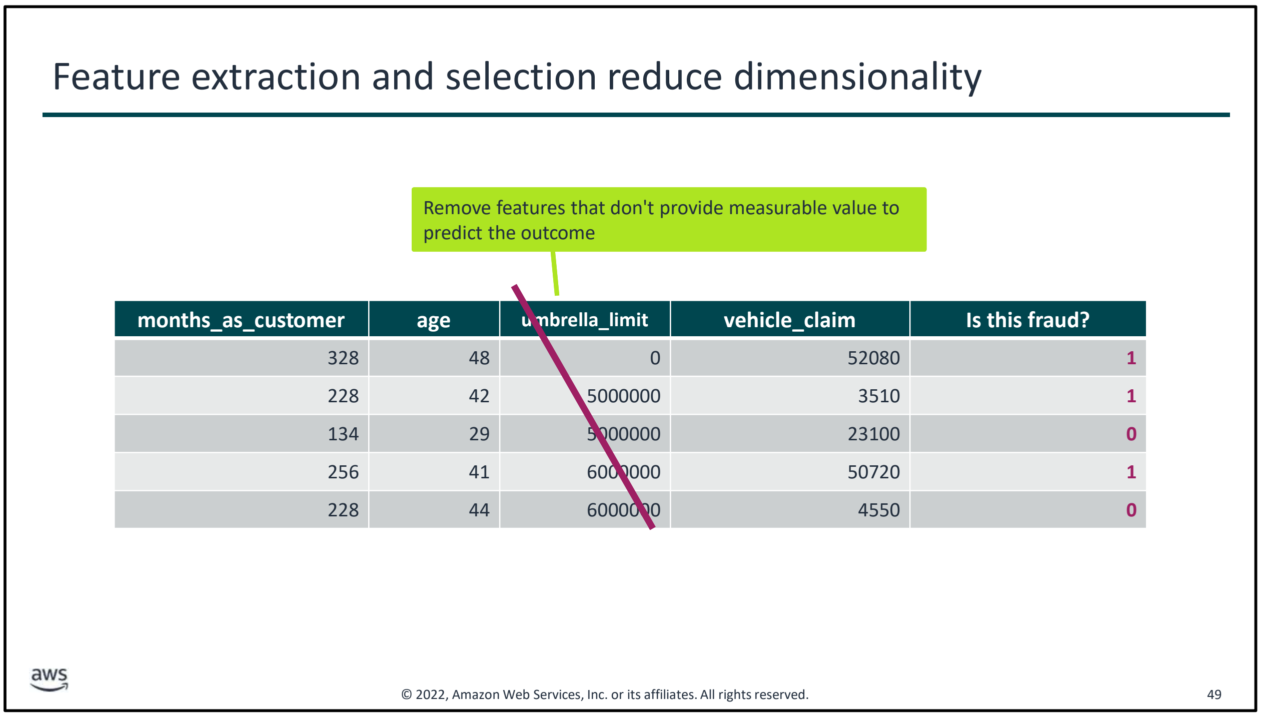
|Instructor notes

|

|Student notes

Feature creation and transformation are both processes that generate new features from existing features by using different types of statistical modeling techniques.

For example, one feature creation technique is called binning. With *binning*, you convert continuous data into groups, also called *bins*. By grouping the items, you provide a fixed number of values to work with, which is easier for the algorithm to work with.



Feature extraction and selection reduce dimensionality

Remove features that don't provide measurable value to predict the outcome

months_as_customer	age	~~umbrella_limit~~	vehicle_claim	Is this fraud?
328	48	~~0~~	52080	1
228	42	~~5000000~~	3510	1
134	29	~~5000000~~	23100	0
256	41	~~6000000~~	50720	1
228	44	~~6000000~~	4550	0

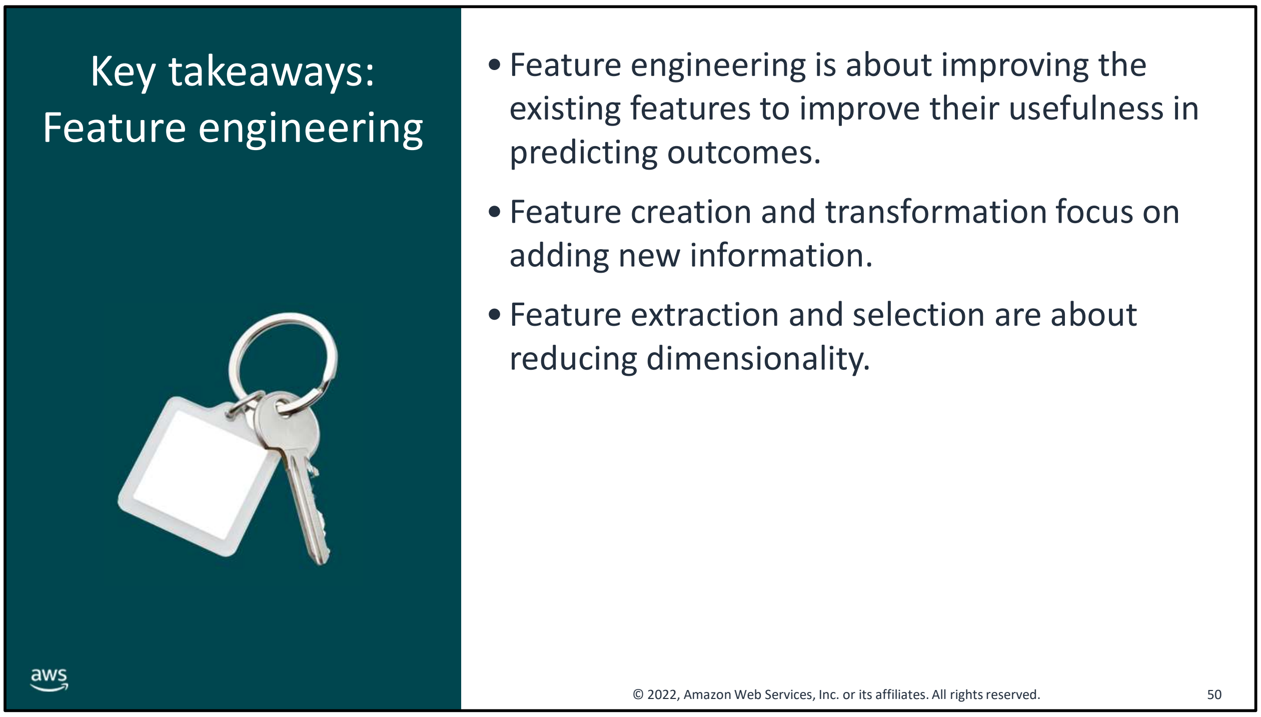
|Slide number 49
|Instructor notes
|
|Student notes

Dimensionality means the number of features that your dataset has. High-dimensional data is data in which the number of features is close to or larger than the number of data points. When a dataset has many different dimensions of data (many features) to sort through, models will have a difficult time finding the patterns that you want them to identify.

Therefore, performing feature extraction and selection is important. An example of feature extraction is removing certain parts of speech, such as articles and prepositions, from a natural language processing dataset.

Feature selection means removing features that don't appear to have a high value in predicting the outcome or that are mostly redundant to another feature that could provide the same input. A domain expert might look at a set of features and be able to identify items that would not be relevant to the business problem.

In the insurance fraud example, the same domain expert who helped to provide the dataset and label the fraudulent items suggested that the umbrella limit is not a particularly useful field when it comes to insurance fraud, so that feature can be removed.



Key takeaways: Feature engineering

- Feature engineering is about improving the existing features to improve their usefulness in predicting outcomes.
- Feature creation and transformation focus on adding new information.
- Feature extraction and selection are about reducing dimensionality.

|Slide number 50
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

The feature engineering step is about using the existing features and improving their value for the model.

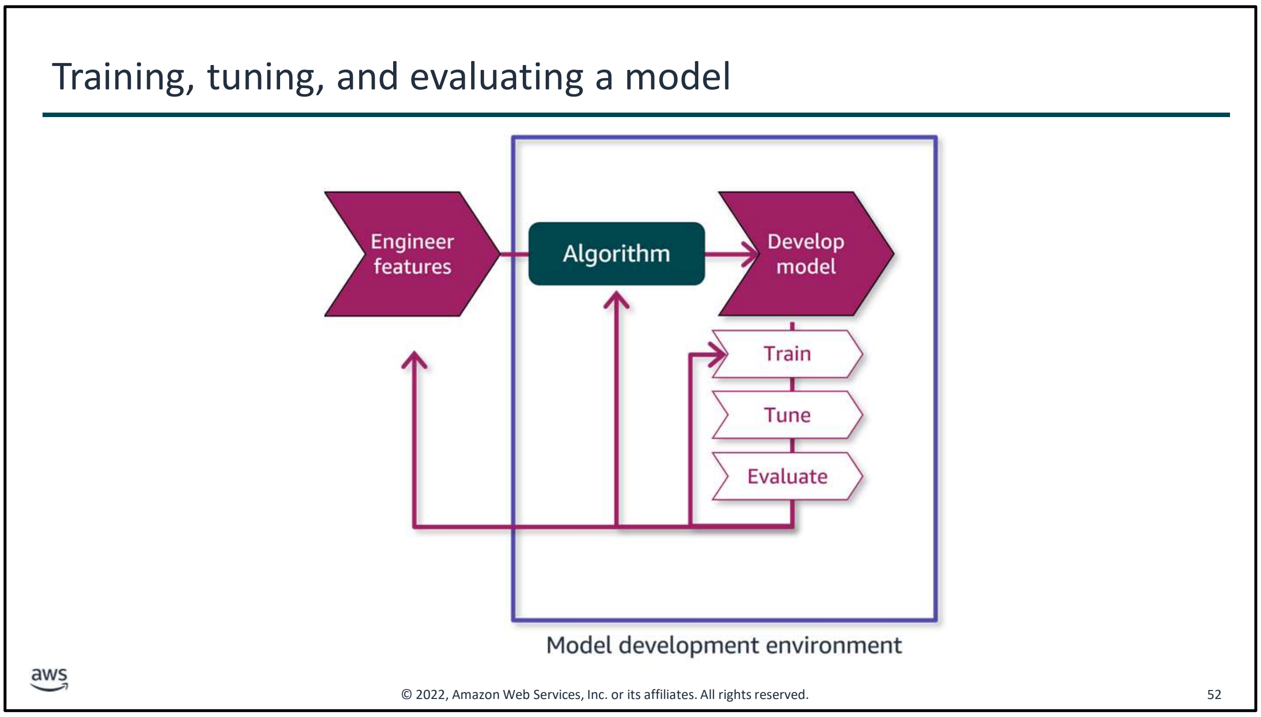
This might include feature creation and transformation to add new values or features.

It might also include feature extraction and selection, which are techniques to reduce the dimensionality of the dataset to make the dataset easier for the model to process.

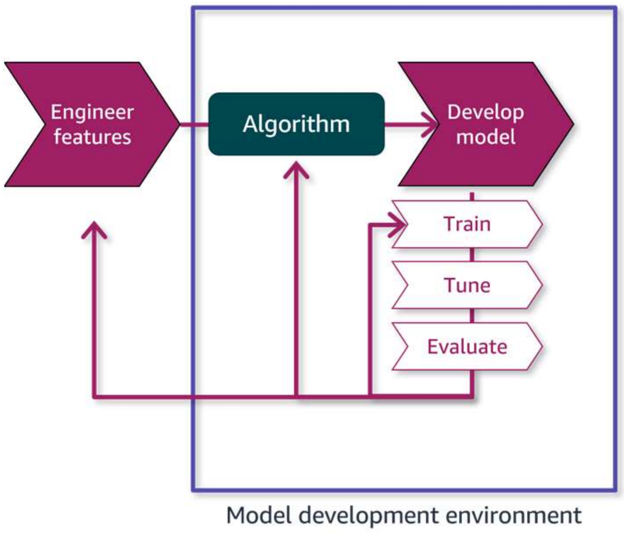
Developing a model

Processing Data for ML

|Slide number 51
|Instructor notes
|
|Student notes
This section describes the develop model phase of the ML lifecycle.



Training, tuning, and evaluating a model

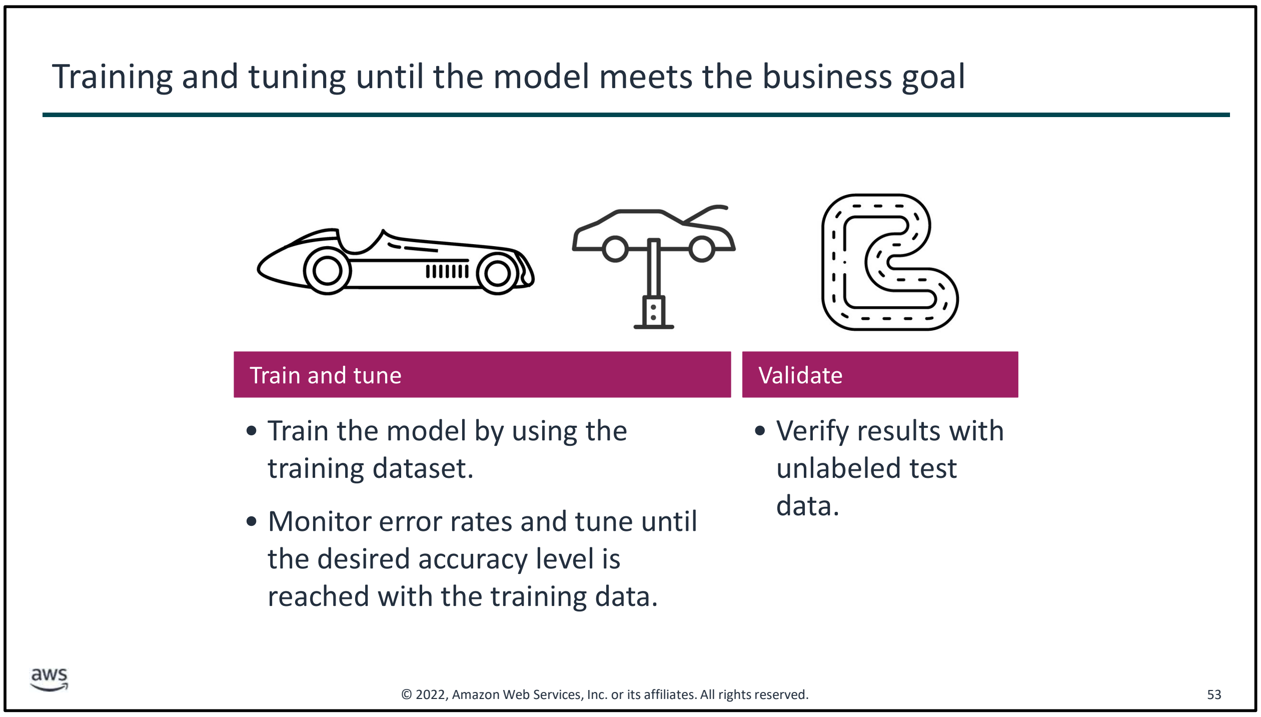


|Slide number 52
|Instructor notes
|
|Student notes

Model development consists of model building, training, tuning, and evaluation. This process is the heart of machine learning.

Model building includes building the algorithm and its supporting code. The code building process should follow best practices that apply to any software development project. For example, your tools should support version control and continuous build, test, and integration of updated code through a pipeline. ML frameworks are available to help developers build and train ML models.

The ML model learns through a repetitive training process by examining features to find patterns, making predictions on those patterns, and evaluating the quality of the predictions. The time and resources that are required to train the model are related to the type of ML model you are running and the dataset that you are using to train the model. For example, a deep learning model would require higher levels of training than a supervised learning model that gives a binary prediction (for example, this is fraud, this is not fraud, or this is a dog, this is not a dog).



Training and tuning until the model meets the business goal

Train and tune

- Train the model by using the training dataset.
- Monitor error rates and tune until the desired accuracy level is reached with the training data.

Validate

- Verify results with unlabeled test data.

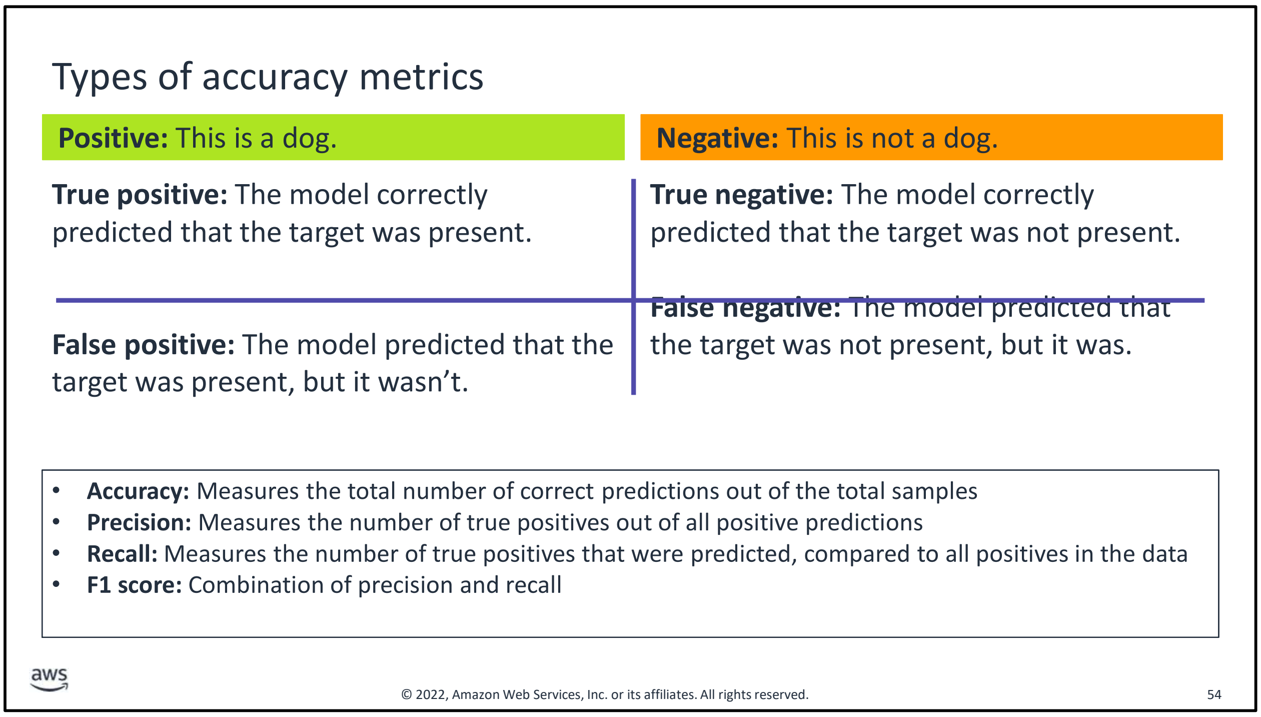
|Slide number 53
|Instructor notes
|
|Student notes

By monitoring errors during the training process, the model improves its understanding of the feature patterns that lead to the correct target values. Each pass through the model allows the tuning of parameters—tweaking the weights of how important the different features are and then seeing if that makes the results better or worse. You measure the error rate of each pass to evaluate the accuracy of the model. Without going into additional detail, note that you might also hear about *hyperparameters*, which are also used to train and test models. *Parameters* are values that you can estimate from the data itself, whereas hyperparameters are external to the model's data. Hyperparameters control the model's learning process.

A data scientist would iterate on training and tuning the model until the error rates indicate that the model is accurate enough for the business problem. For example, the business might determine that the predictions need to be correct 80 percent of the time to meet their goals.

After you have tuned your model to the accuracy level that is expected for the business problem, it’s time to run the model with the test dataset to validate the model's accuracy beyond the training dataset. For testing, you remove the known label off of the test dataset, feed the test dataset through the model, and evaluate its predicted values compared to the actual label values.

Based on the results of training, tuning, and evaluating the model with test data, the data scientist might fine-tune the data, the algorithm, or both. If the results are acceptable, the model is ready to be deployed.



Types of accuracy metrics

Positive: This is a dog.	**Negative:** This is not a dog.
True positive: The model correctly predicted that the target was present.	**True negative:** The model correctly predicted that the target was not present.
False positive: The model predicted that the target was present, but it wasn't.	**False negative:** The model predicted that the target was not present, but it was.

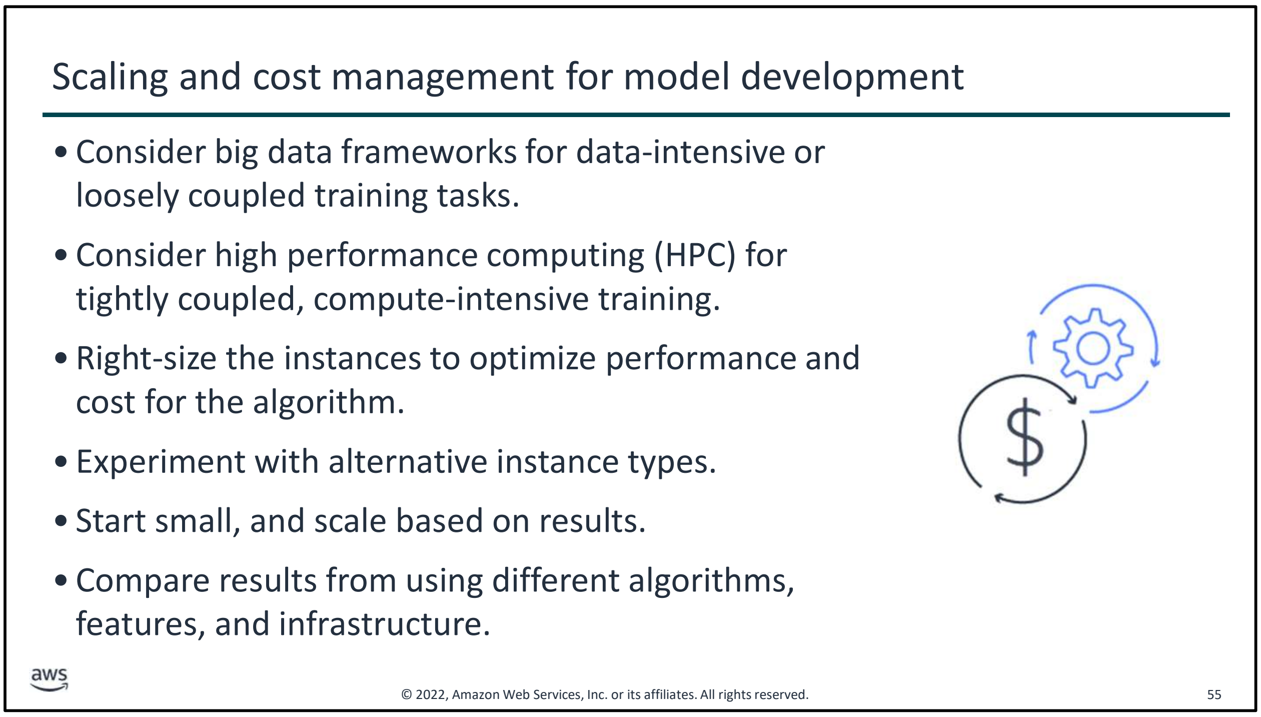
- **Accuracy:** Measures the total number of correct predictions out of the total samples
- **Precision:** Measures the number of true positives out of all positive predictions
- **Recall:** Measures the number of true positives that were predicted, compared to all positives in the data
- **F1 score:** Combination of precision and recall

~Script notes: Append A true positive is when the model predicts that the target exists in the data and is correct. For example, the model says "this is a dog" when the picture contains a dog. A false positive would be when the model predicts there is a dog but there isn't. A true negative means the model correctly predicted that the target is not present—this is not a dog—and a false negative would mean that the model did not identify the target but should have.
~Precision is a measure of how many true positives were predicted out of all the positive predictions that the model made. For example, your model made 100 positive predictions about a dog being in the picture, but only 50 of them were actually dogs.
~Recall measures how many times the model predicted true positives compared to how many actual positives exist in the data. For example, maybe your model got 50 true positives on dogs but also missed 150 dogs that it should have predicted.
~The F1 score is a combination of these two approaches and is often used as a quick validation of a model.
~
|Slide number 54
|Instructor notes
|
|Student notes
The actual metrics and methods for tuning the model are much more complex than

described in this course, but here are a few general metrics that you should be familiar with.

Accuracy is the base metric that is used to evaluate a model and is the measure of the number of correct predictions when compared to all predictions made.

As noted on the slide, other measures incorporate data about how many times the model positively predicted the presence of a target and how many times it predicted the absence of the target.



Scaling and cost management for model development

- Consider big data frameworks for data-intensive or loosely coupled training tasks.
- Consider high performance computing (HPC) for tightly coupled, compute-intensive training.
- Right-size the instances to optimize performance and cost for the algorithm.
- Experiment with alternative instance types.
- Start small, and scale based on results.
- Compare results from using different algorithms, features, and infrastructure.

|Slide number 55
|Instructor notes
|The sections on AWS ML services provide examples of how these considerations are reflected in services available for ML applications. For example, Amazon EMR for big data processing and AWS ParallelCluster for high performance computing (HPC).
|
|Student notes
The process of building, training, and tuning the model is resource intensive, and the model development environment requires an infrastructure that is designed for ML processing. To train and tune your model, you need compute, networking, and storage resources. In addition to the characteristics this course discusses, such as the volume, velocity, and type of data to be processed, the complexity of the model and the algorithm that you use will also influence what type of infrastructure you will need. Choosing GPU-based instances can help you train models faster, which enables data scientists to iterate faster, train more models, and increase accuracy.

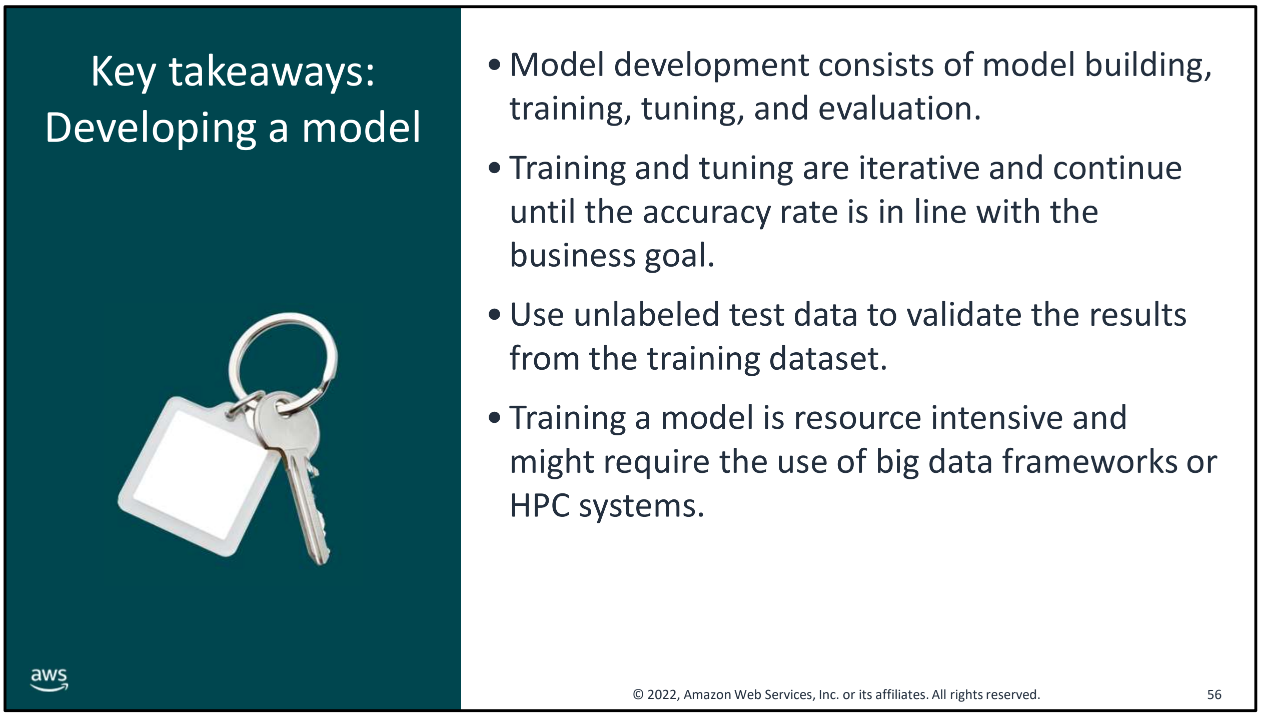
Training activities often run on big data frameworks, such as those described in the Processing Big Data module. Clustered big data configurations provide high-volume parallel processing of a lot of data at one time and are good for calculations that can be split up and worked on in pieces (loosely coupled). An example of an ML problem that might take this approach is genomics. In genomic ML, a researcher might iterate on a dataset, tweaking parameters and optimizing thousands of times. For this type of work, a big data solution can speed up the parallel processing of those iterations.

Another cluster-based configuration that is often used for training ML models is high performance computing (HPC). HPC is also a clustered solution, but it is optimized for compute-intensive operations, where the computations are tightly coupled, meaning the work cannot be distributed into individual tasks. With HPC, the single computation can be spread across many instances in a cluster. A weather researching who is developing a model of weather across the United States would use an HPC system for ML models.

But bigger is not always better. Simple models might not train faster on larger instances because they might not be able to benefit from increased parallelism. These models might even train slower, because of the high communication overhead of GPUs.

The ML infrastructure on AWS section describes the types of infrastructure choices and services that can help you match resources to the scale of the training that you need to perform and the costs that you are trying to manage. As a best practice, start with smaller instances and scale as necessary.

It is a best practice to evaluate and compare how your model performs with different algorithms, features, and infrastructure.



Key takeaways: Developing a model

- Model development consists of model building, training, tuning, and evaluation.
- Training and tuning are iterative and continue until the accuracy rate is in line with the business goal.
- Use unlabeled test data to validate the results from the training dataset.
- Training a model is resource intensive and might require the use of big data frameworks or HPC systems.

|Slide number 56
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

Model development is the heart of machine learning.

The model is trained and tuned until it reaches the desired accuracy level.

Test data is then used to validate results with a subset of data that is not labeled.

You need to right-size your resources for the training phase, which is resource intensive. Often, big data frameworks or HPC systems are used to train models.

Deploying a model

Processing Data for ML

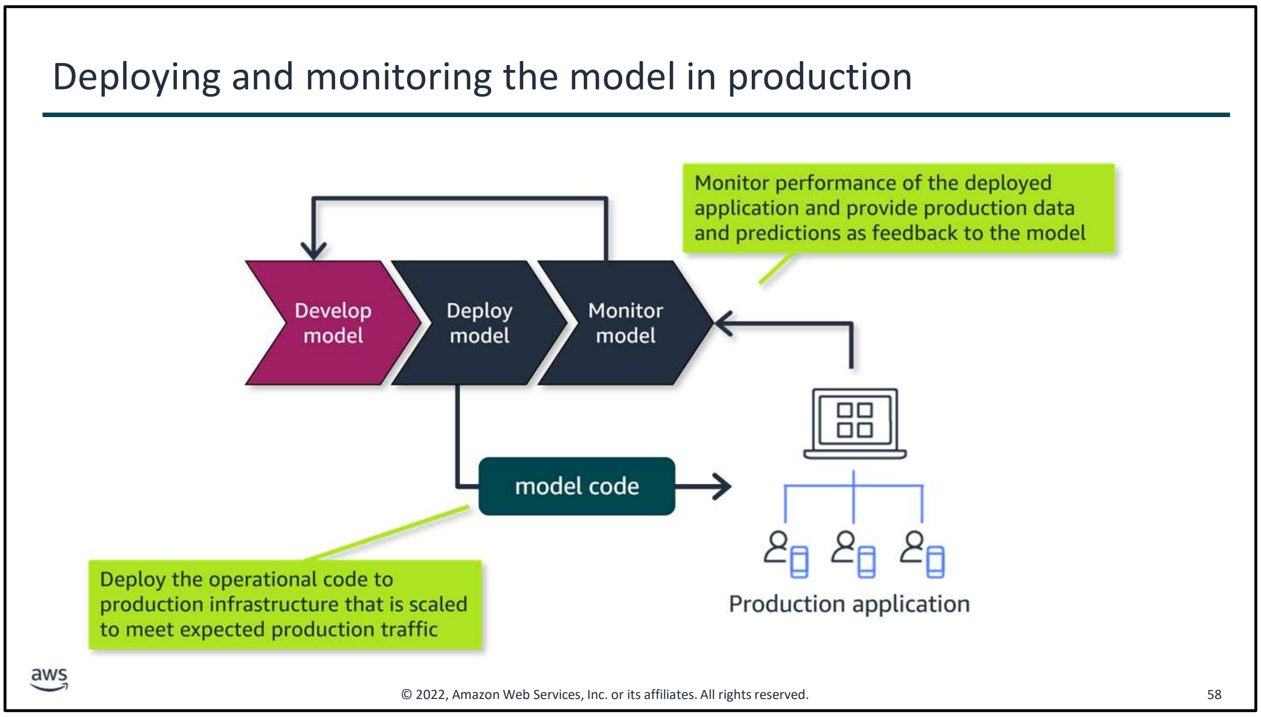
|Slide number 57

|Instructor notes

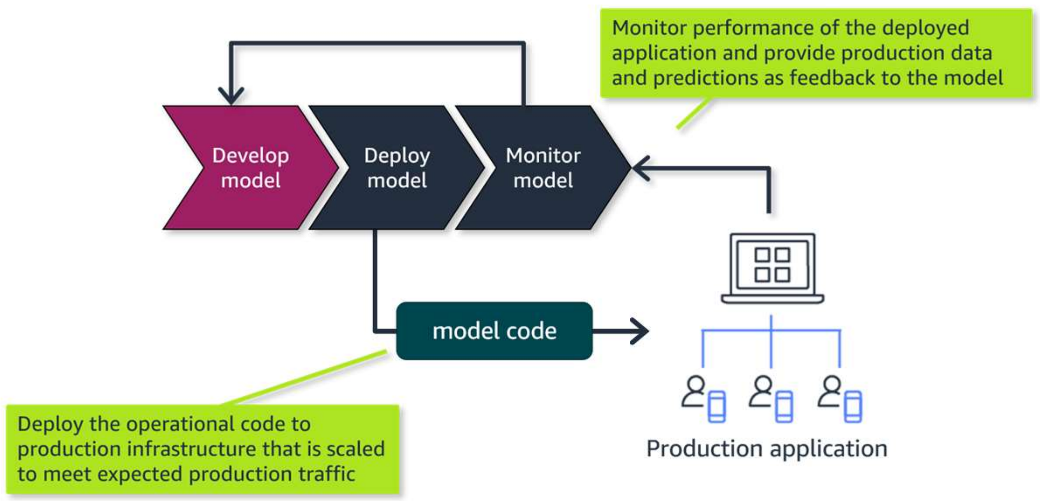
|

|Student notes

This section describes the deploy model phase of the ML lifecycle.



Deploying and monitoring the model in production

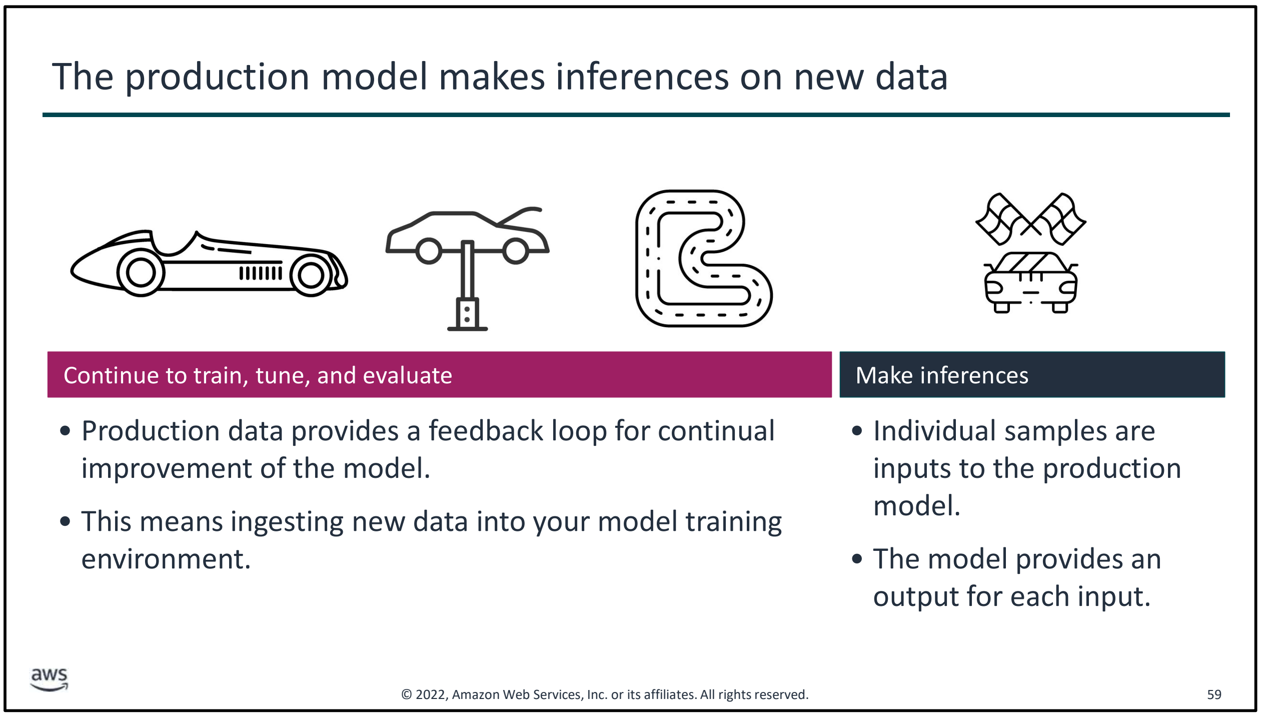

Monitor performance of the deployed application and provide production data and predictions as feedback to the model
Develop model
Deploy model
Monitor model
model code
Production application
Deploy the operational code to production infrastructure that is scaled to meet expected production traffic

|Slide number 58
|Instructor notes
|
|Student notes
After you have trained and optimized your model to your desired level of accuracy and precision, you put it into production to make predictions.

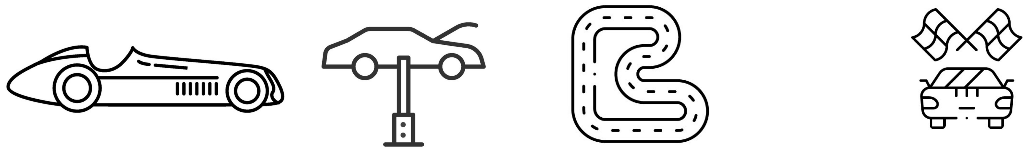
Standard debugging and logging of the application are part of monitoring the deployed model. You want to capture normal application metrics and track items such as slow queries, high latency, and problem events.

However, monitoring the model's outputs in production is critical to ensure that the model is producing the expected level of accuracy with production data. By monitoring the model, you can also adapt to changes in data patterns that might cause the model to become less accurate over time. From a purely operational perspective, there might not be errors, but this doesn't mean that the model is performing as it needs to.

Typically, roles that differ from those who ingested and prepared the data, or trained and tuned the model, will deploy the model into production.



The production model makes inferences on new data



Continue to train, tune, and evaluate

- Production data provides a feedback loop for continual improvement of the model.
- This means ingesting new data into your model training environment.

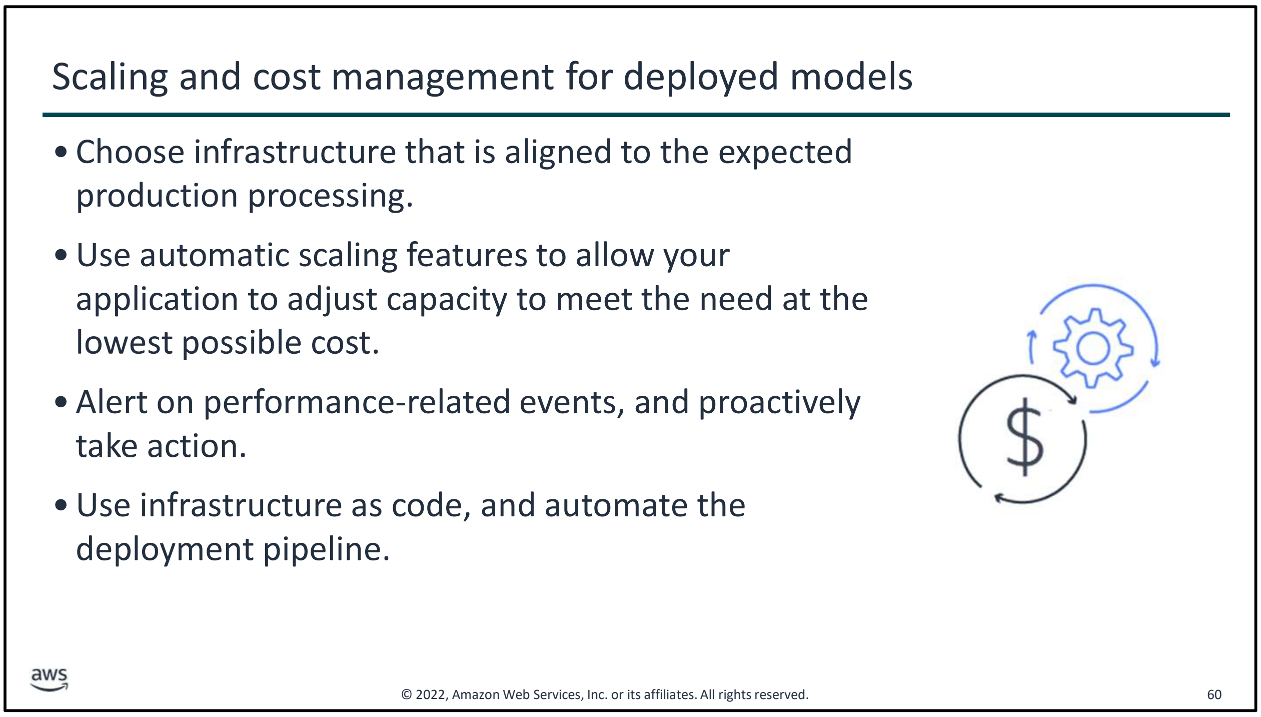
Make inferences

- Individual samples are inputs to the production model.
- The model provides an output for each input.

|Slide number 59
|Instructor notes
|
|Student notes

Inference is the process of making predictions on the production data. The code and infrastructure that are related to production are referred to as inference.

Notably, in production, the model takes in individual samples and provides an output. Compare this to the training environment, where the model churns through high volumes of transactions in parallel.

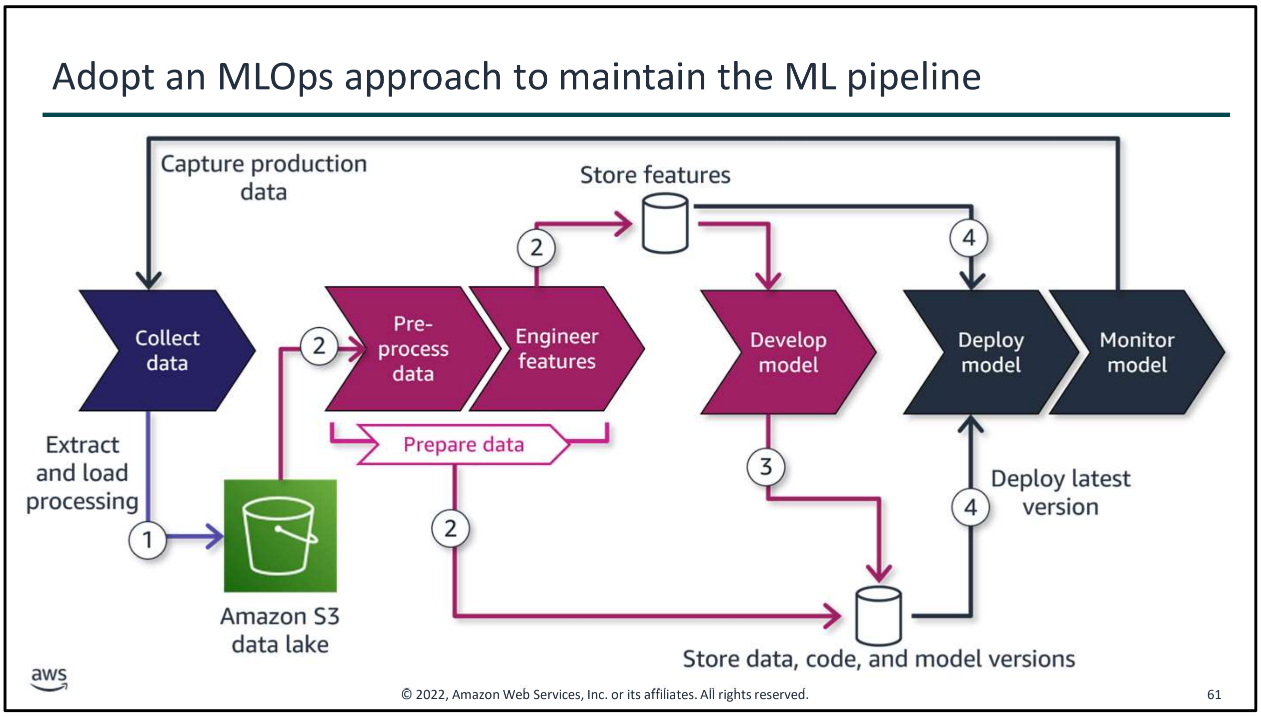


Scaling and cost management for deployed models

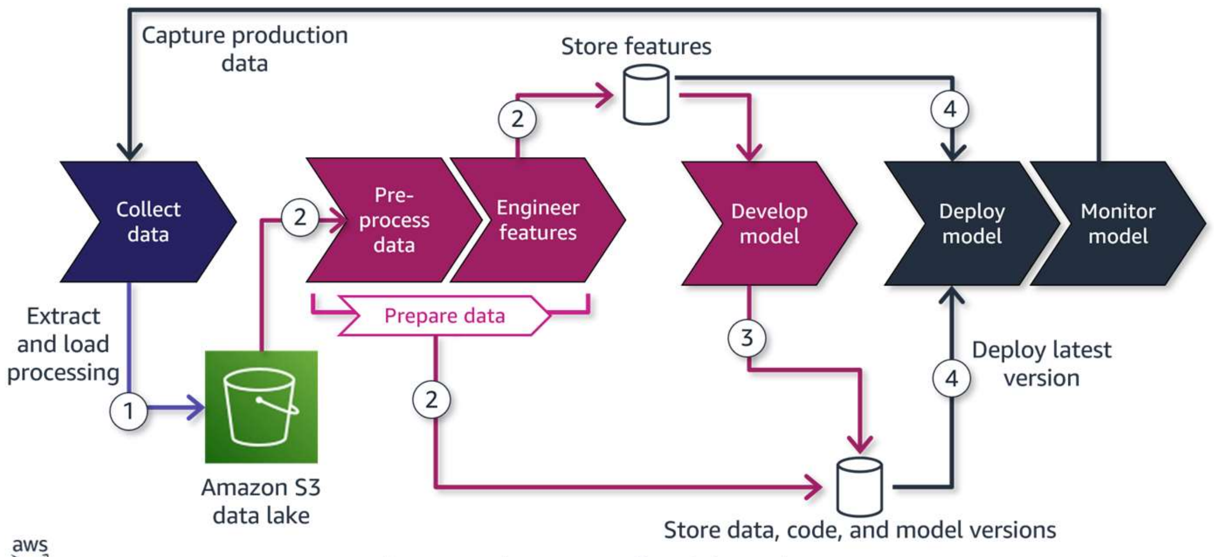
- Choose infrastructure that is aligned to the expected production processing.
- Use automatic scaling features to allow your application to adjust capacity to meet the need at the lowest possible cost.
- Alert on performance-related events, and proactively take action.
- Use infrastructure as code, and automate the deployment pipeline.

|Slide number 60
|Instructor notes
|The sections on AWS ML services provide examples of how these considerations are reflected in services available for ML applications.
|
|Student notes
As with any application, you want to ensure that it performs as expected in production while managing the costs to operate it. Importantly, with ML applications, the choices that you need to make to train the model differ greatly from those to put it into production. Training jobs batch process hundreds of data samples in parallel, and inference jobs usually process a single input in real time. Therefore, training the model typically requires much higher levels of compute power than the deployed model.

Aside from selecting resources to deploy the model, scaling your ML solution means building your environments with infrastructure as code and automating the ML deployment pipeline. You want to be able to respond quickly to changes without a lot of human intervention.



Adopt an MLOps approach to maintain the ML pipeline



|Slide number 61
|Instructor notes
|
|Student notes

This module has looked at each of these steps on its own, and each of them might be performed by different teams or team members who use different resources and infrastructure. But when you need to update the ML solution, you need to reflect those updates across all the integrated steps. The four different sets of activities that are numbered in the diagram highlight flows that are involved in an update.

In this example, if you want to incorporate additional data into your training dataset and retune the model, you would need to initiate the following flows:

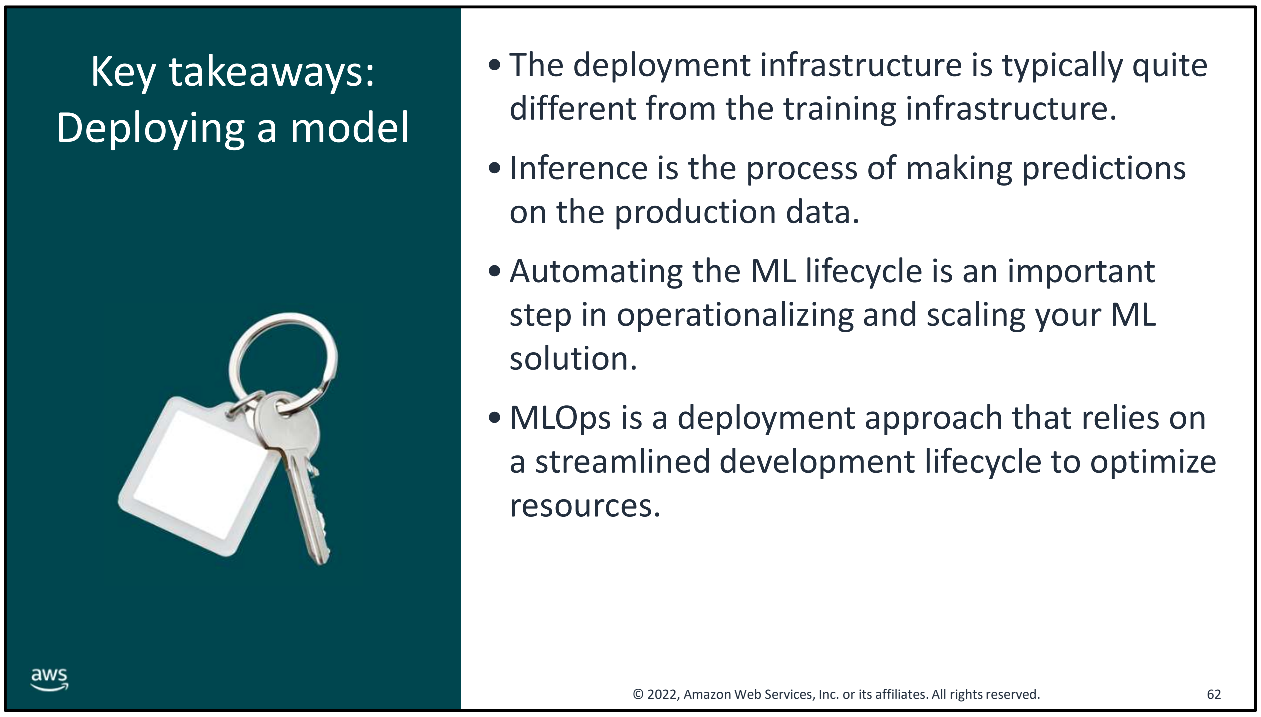
1. The data engineer extracts and loads the additional data into the data lake.
2. The data scientist brings the new data into the processing phase and then saves any updates to the data and the features.
3. The data scientist then trains and tunes the model by using the updated features and data and saves the updated model version.
4. The ML operations team retrieves the latest model and deploys the latest code to the production environment.

Each process relies on different sets of infrastructure that someone or some team is responsible for maintaining, optimizing, and securing access to. This includes the resources that are required to extract and load the data into the data lake, the processing environment where the data is visualized and prepared for training, the model development environment where the model is trained, and the production environment where the model is deployed. Depending on how your organization assigns these responsibilities, deploying updates to an ML solution can quickly become complex to manage manually. Automation of these processes not only streamlines updates but reduces the chance of human error across these integrated activities.

DevOps is the combination of cultural engineering practices and patterns, and tools that increase an organization's ability to deliver applications and services at high velocity and better quality. Over time, several essential practices have emerged when adopting DevOps: continuous integration and continuous delivery (CI/CD), infrastructure as code, and monitoring and logging.

Like DevOps, MLOps relies on a collaborative and streamlined approach to the ML development lifecycle. The intersection of people, process, and technology optimizes the start-to-finish activities that are required to develop, build, and operate ML workloads. MLOps focuses on the intersection of data science and data engineering in combination with existing DevOps practices to streamline model delivery across the ML development lifecycle. MLOps requires the integration of software development, operations, data engineering, and data science.

The Automating the Pipeline module goes into greater detail about tools and processes to automate data pipelines. The key takeaway is to recognize that you want the interrelated processes of ingesting data into your model, training and tuning the model, and deploying the model to be automated.



Key takeaways: Deploying a model

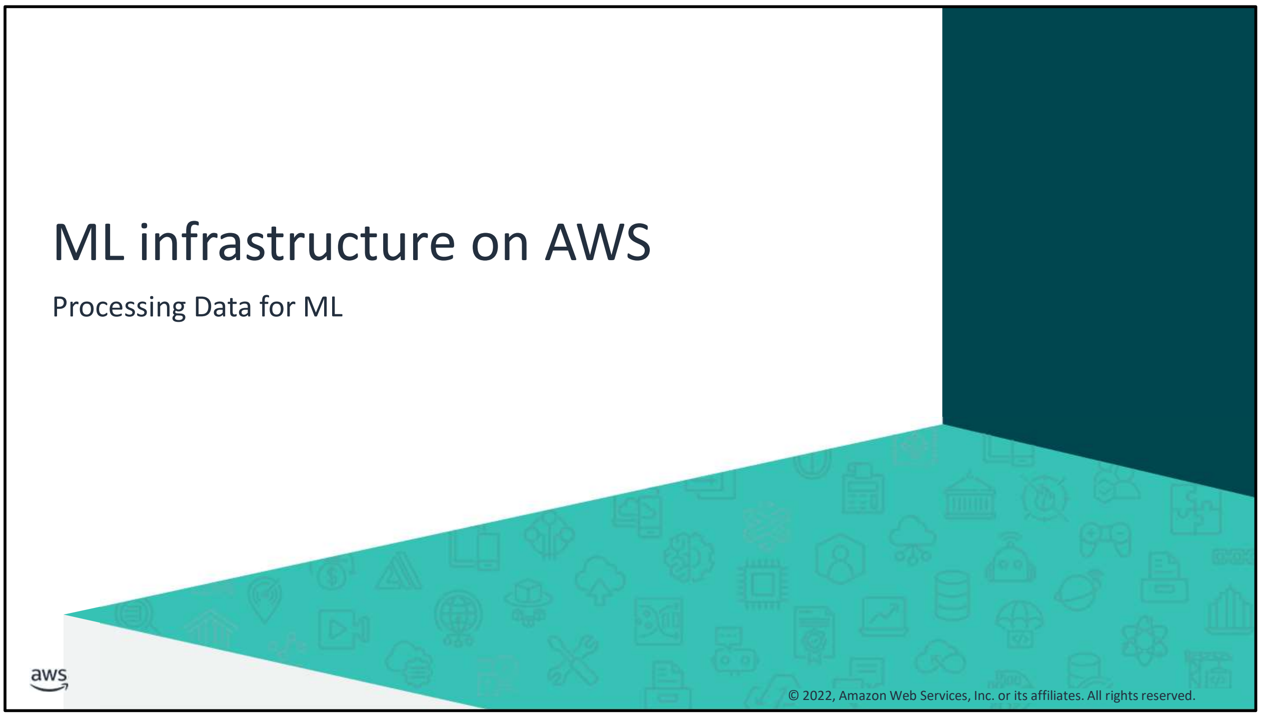
- The deployment infrastructure is typically quite different from the training infrastructure.
- Inference is the process of making predictions on the production data.
- Automating the ML lifecycle is an important step in operationalizing and scaling your ML solution.
- MLOps is a deployment approach that relies on a streamlined development lifecycle to optimize resources.

|Slide number 62
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

When the model is deployed into production, the infrastructure typically has different characteristics than the training environment.

The code and infrastructure that are related to the production model are referred to as inference.

The large number of roles and different environments that are part of the ML lifecycle need an automated deployment strategy. MLOps is a best practice that provides an approach to create a collaborative and streamlined approach to deployments.

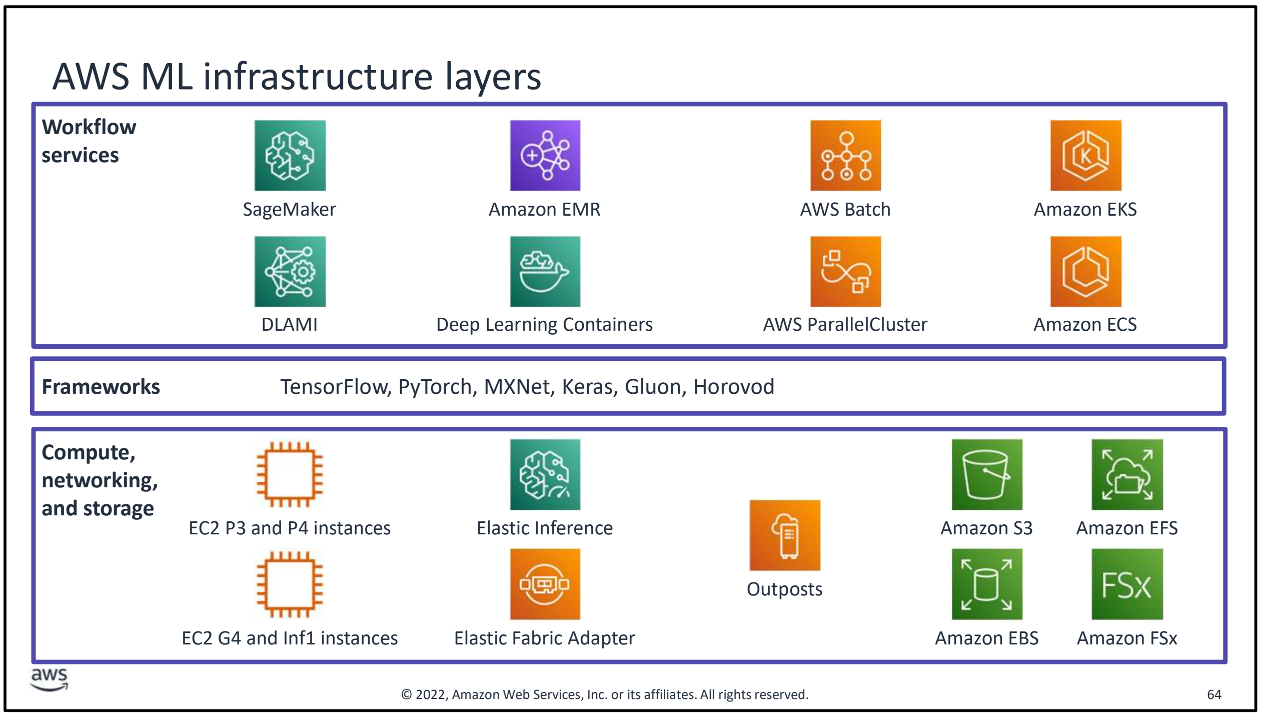


ML infrastructure on AWS

Processing Data for ML

|Slide number 63
|Instructor notes
|This section and the two that follow illustrate the increasing levels of abstraction that are available in the cloud to develop ML solutions.
|First, services are available to provide the compute, storage, and networking that are designed to handle the high demands of ML training and production inference. An organization that uses these services for ML still needs a lot of expertise in architecting and deploying ML models that scale to meet processing demands. Workflow services are available to alleviate some of the burden of managing containers, big data processing, and HPC distributed environments. This section describes those services, which can generally map back to the scaling and cost management slides for training and deployed models that were presented earlier.
|The next section discusses Amazon SageMaker and illustrates how this type of service can further reduce the burden of managing ML infrastructure by providing managed environments and integrated tools across the ML lifecycle.
|The final section gives examples of purpose-built services that are designed to make it much easier to perform common ML applications, such as natural language processing or image processing. The theme is a recurring one for cloud-based solutions. Each step higher on the infrastructure spectrum (from on-premises servers to virtualized servers, to cloud-based services, to managed and serverless cloud-based services) provides an organization

with the ability to choose how much of the underlying infrastructure they want to control and manage, balanced against how quickly they can get started and experiment with ML without as much overhead.
|
|Student notes
This section describes the AWS services that provide an ML infrastructure.



AWS ML infrastructure layers

64

~ https://aws.amazon.com/machine-learning/infrastructure/

|**Instructor notes**

|The full graphic for this is on the AWS Machine Learning Infrastructure page. The link is included in the course resources. The diagram on the slide has been reduced to focus on the types of things in each layer relevant to the content in this module.

|

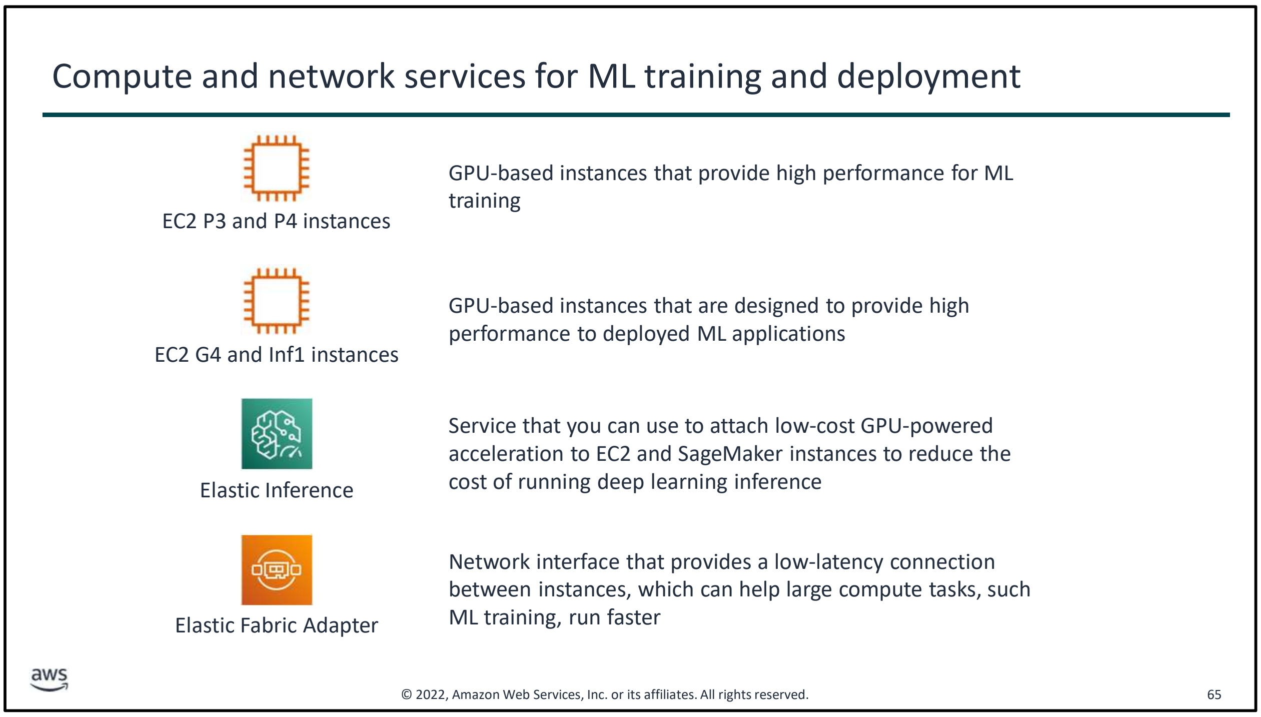
|**Student notes**

The slide graphic illustrates three layers of ML services offered at AWS, but you could compare the type of functionality and features addressed at each layer with other cloud providers. The key is to understand the types of tools that each layer provides and to determine the appropriate combination for your use case, team members, and organization. Some services are designed specifically for machine learning, including Amazon SageMaker, AWS Deep Learning AMIs (DLAMI), AWS Deep Learning Containers, and Amazon Elastic Inference. Other services work well for common ML requirements, including big data or HPC computing; GPU-based compute; and low cost, fast storage.

The bottom layer shows examples of compute, networking, and storage services that make up the foundational blocks of ML infrastructure. If you choose to build and manage your own infrastructure, these are the types of services that you would use to host your ML infrastructure. The middle layer highlights ML frameworks that the AWS ML infrastructure supports. These frameworks make it easier for developers to build and deploy ML models.

The top layer contains workflow services. These make it easier for you to manage and scale your underlying ML infrastructure by abstracting the complexity of setting up your infrastructure and by providing integrations that simplify movement through the ML lifecycle.

The next slides look in a bit more detail at key services that are related to this course. SageMaker is discussed in the section after this one. A link to the AWS Machine Learning Infrastructure page is provided in the course resources.



Compute and network services for ML training and deployment

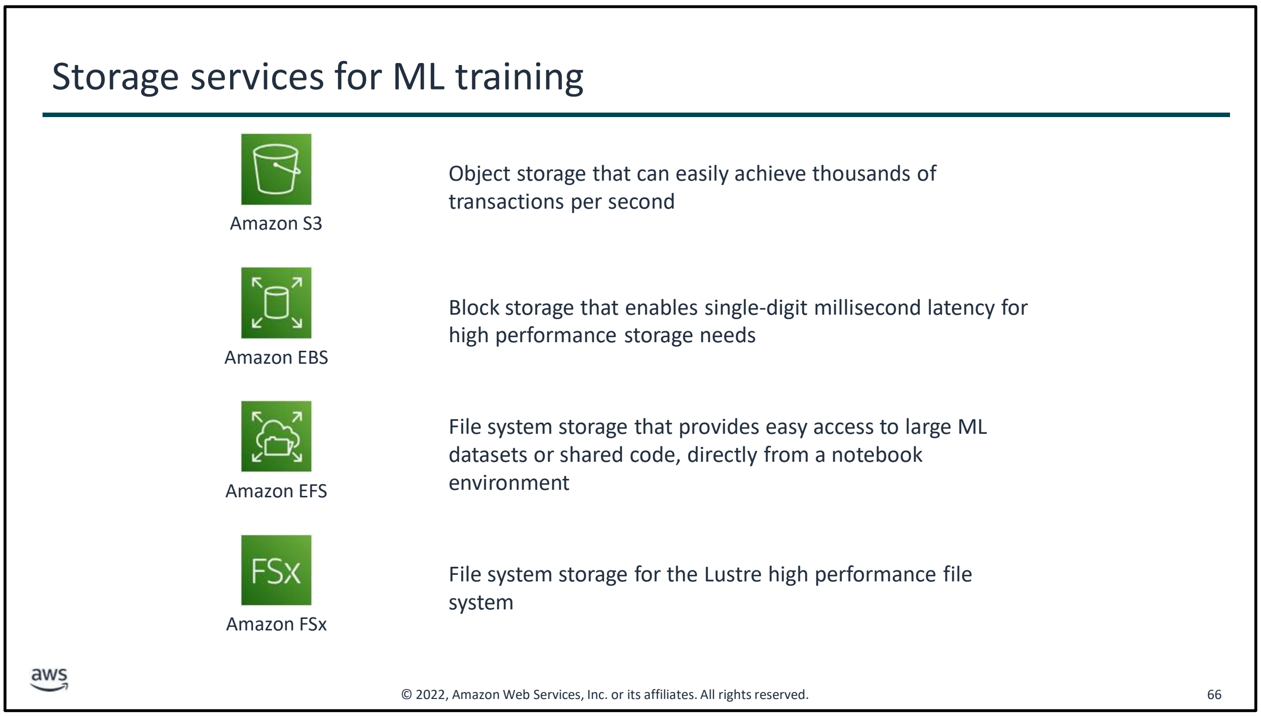
EC2 P3 and P4 instances	GPU-based instances that provide high performance for ML training
EC2 G4 and Inf1 instances	GPU-based instances that are designed to provide high performance to deployed ML applications
Elastic Inference	Service that you can use to attach low-cost GPU-powered acceleration to EC2 and SageMaker instances to reduce the cost of running deep learning inference
Elastic Fabric Adapter	Network interface that provides a low-latency connection between instances, which can help large compute tasks, such ML training, run faster

~ https://aws.amazon.com/machine-learning/infrastructure/
|Slide number 65
|Instructor notes
|
|Student notes
The services that are described here all help to provide the compute power and speed that are needed for ML applications.

Amazon Elastic Compute Cloud (Amazon EC2) has instance types that are especially suitable to ML applications. The P3 and P4 types are GPU based and provide high-performance compute to train ML models. The G4 and Inf1 instance types are well suited to running inference on deployed models.

With the Elastic Inference service, you can attach low-cost GPU-powered acceleration to other EC2 instance types to reduce the cost of running deep learning inference.

Elastic Fabric Adapter is a network interface that provides a low-latency connection between instances in a clustered configuration. This can help ML models to train faster.

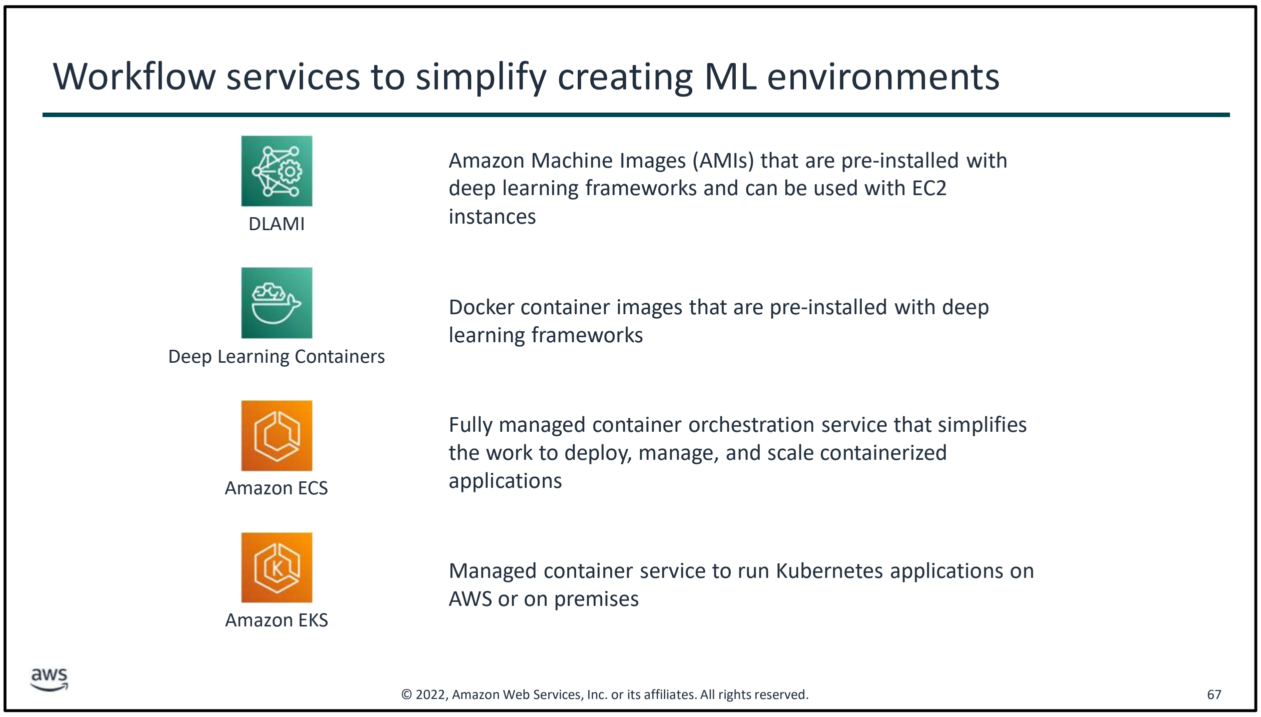


Storage services for ML training

Amazon S3	Object storage that can easily achieve thousands of transactions per second
Amazon EBS	Block storage that enables single-digit millisecond latency for high performance storage needs
Amazon EFS	File system storage that provides easy access to large ML datasets or shared code, directly from a notebook environment
Amazon FSx	File system storage for the Lustre high performance file system

~Script: Remove the “Service names” part from the student notes.
~ https://aws.amazon.com/machine-learning/infrastructure/
|Slide number 66
|Instructor notes
|
|Student notes
The storage services that are described here provide for the size, type, and speed of processing that are required to develop models.

Amazon EBS is Amazon Elastic Block Store and Amazon EFS is Amazon Elastic File System



Workflow services to simplify creating ML environments

DLAMI	Amazon Machine Images (AMIs) that are pre-installed with deep learning frameworks and can be used with EC2 instances
Deep Learning Containers	Docker container images that are pre-installed with deep learning frameworks
Amazon ECS	Fully managed container orchestration service that simplifies the work to deploy, manage, and scale containerized applications
Amazon EKS	Managed container service to run Kubernetes applications on AWS or on premises

~ https://aws.amazon.com/machine-learning/infrastructure/

|Slide number 67

|Instructor notes

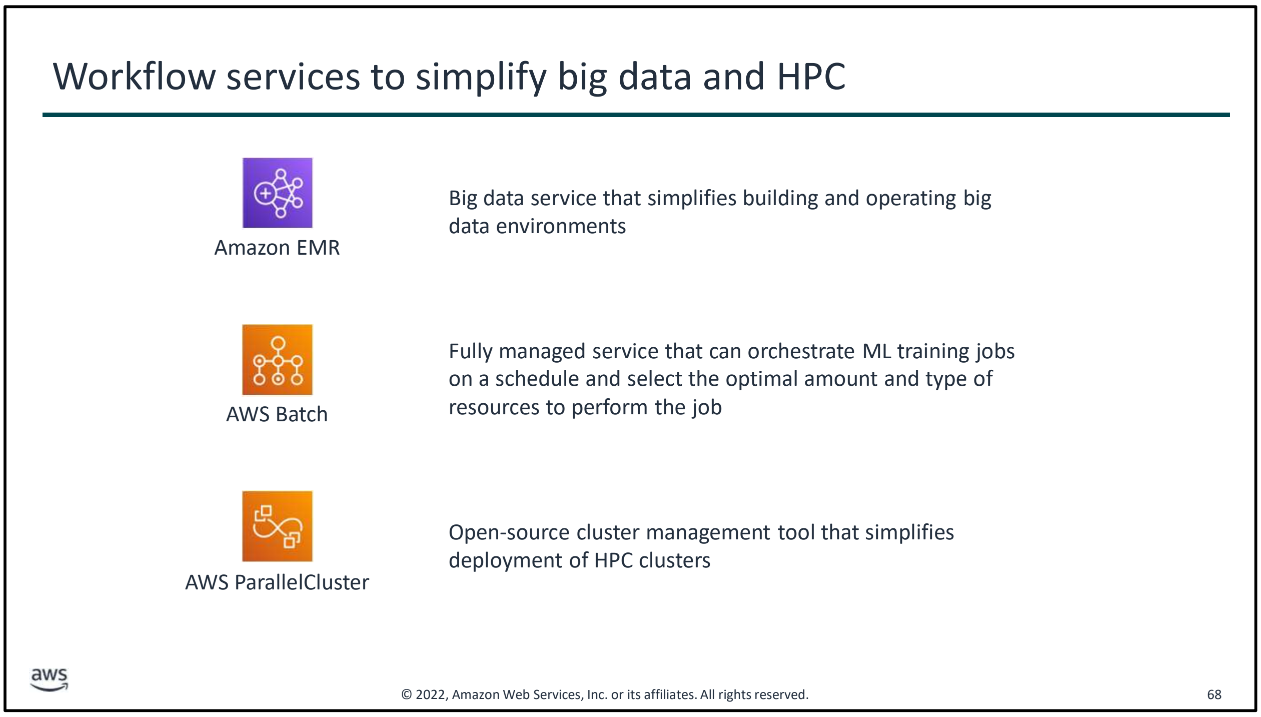
|

|Student notes

These services from the workflow services layer help to simplify the work that is required to build an ML environment.

The first two are specifically designed for deep learning.

Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS) are two options for managing container deployments. Kubernetes is an open-source system to automate deployment, scaling, and management of containerized applications.



Workflow services to simplify big data and HPC

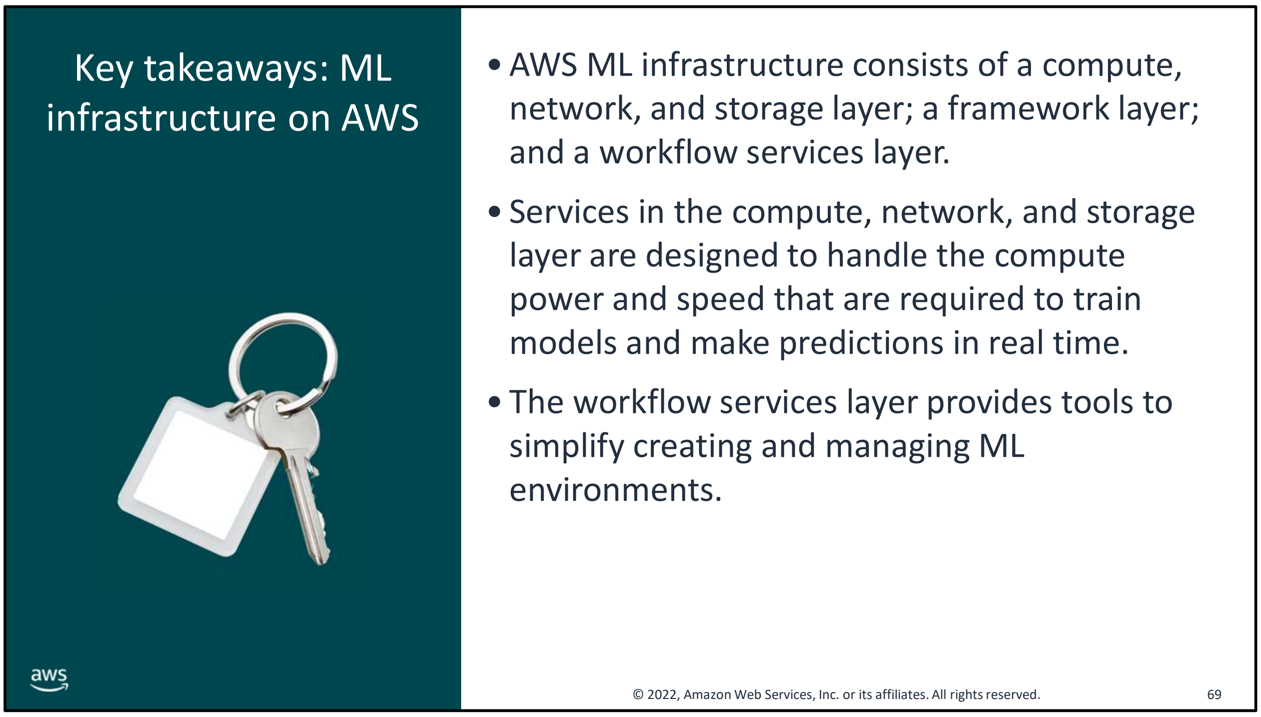
Amazon EMR	Big data service that simplifies building and operating big data environments
AWS Batch	Fully managed service that can orchestrate ML training jobs on a schedule and select the optimal amount and type of resources to perform the job
AWS ParallelCluster	Open-source cluster management tool that simplifies deployment of HPC clusters

~ https://aws.amazon.com/machine-learning/infrastructure/
|Slide number 68
|Instructor notes
|
|Student notes
These services from the workflow services layer help to simplify building a big data or HPC system for ML model training.

Amazon EMR is covered in detail in the Processing Big Data module.

AWS Batch is an orchestration service that can run complex training jobs on a schedule. The service chooses just enough compute resources to run the job and then shuts them down when the job is complete.

AWS ParallelCluster is an open-source cluster management tool that simplifies deployment of HPC clusters.



Key takeaways: ML infrastructure on AWS

- AWS ML infrastructure consists of a compute, network, and storage layer; a framework layer; and a workflow services layer.
- Services in the compute, network, and storage layer are designed to handle the compute power and speed that are required to train models and make predictions in real time.
- The workflow services layer provides tools to simplify creating and managing ML environments.

|Slide number 69
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

The AWS ML infrastructure services are grouped into three layers, with the compute, networking, and storage at the bottom and workflow services at the top. In the middle are popular frameworks that are integrated with the AWS ML tools. The workflow services help to simplify creating and managing ML environments.

SageMaker

Processing Data for ML

|Slide number 70
|Instructor notes
|
|Student notes
This section introduces key features of Amazon SageMaker.

SageMaker provides an integrated workbench for ML

Integrated features and tools include the following:

- SageMaker Studio is a web-based visual interface where you can perform all ML development steps.
- SageMaker Data Wrangler simplifies the process of data preparation and feature engineering.
- SageMaker Processing gives you a simplified experience to run ML processing steps.
- SageMaker Canvas provides business analysts with a visual point-and-click interface that to generate accurate ML predictions on their own.

Amazon SageMaker

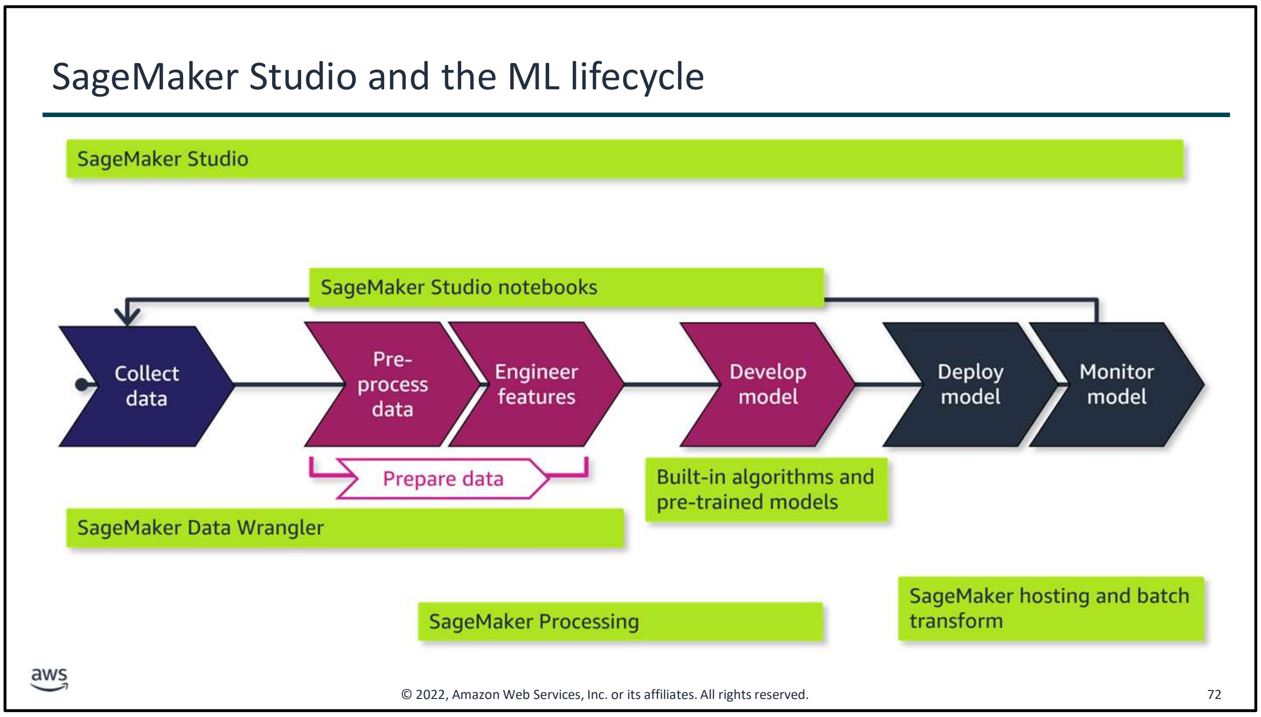
|Slide number 71

|Instructor notes

|

|Student notes

SageMaker includes additional features and tools that you can explore for ML projects.



SageMaker Studio and the ML lifecycle

|Slide number 72
|Instructor notes
|
|Student notes

Amazon SageMaker Studio provides an integrated development environment (IDE) for ML from which you can build, train, deploy, and analyze models—all from the same application.

Amazon SageMaker Data Wrangler is a feature of SageMaker Studio, and you can use it to ingest data, explore it, perform preprocessing steps, and perform feature engineering. The tool includes standard transforms for cleaning and feature engineering and can automatically generate insights about data quality and abnormalities. You can also define a workflow of wrangling steps to be performed. This provides a low-code way to do a lot of the common data preparation steps.

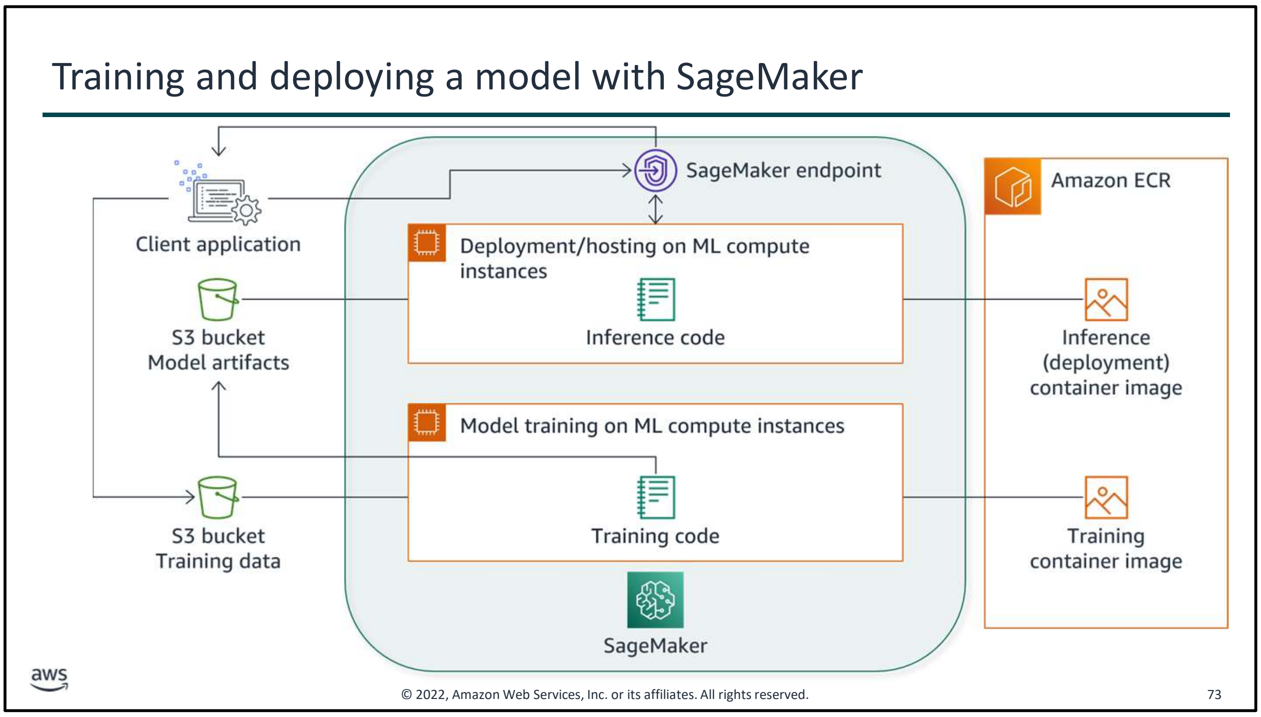
The Processing Big Data module introduces the concept of coding using Jupyter notebooks. SageMaker Studio notebooks are collaborative notebooks that you can launch quickly because you don't need to set up compute instances and file storage in advance. SageMaker Studio notebooks provide persistent storage, which enables you to view and share notebooks even if the instances that the notebooks run on are shut down. A SageMaker Studio notebook runs in a managed environment that includes a configurable

EC2 instance and a container image with the files that are needed to run the notebook.

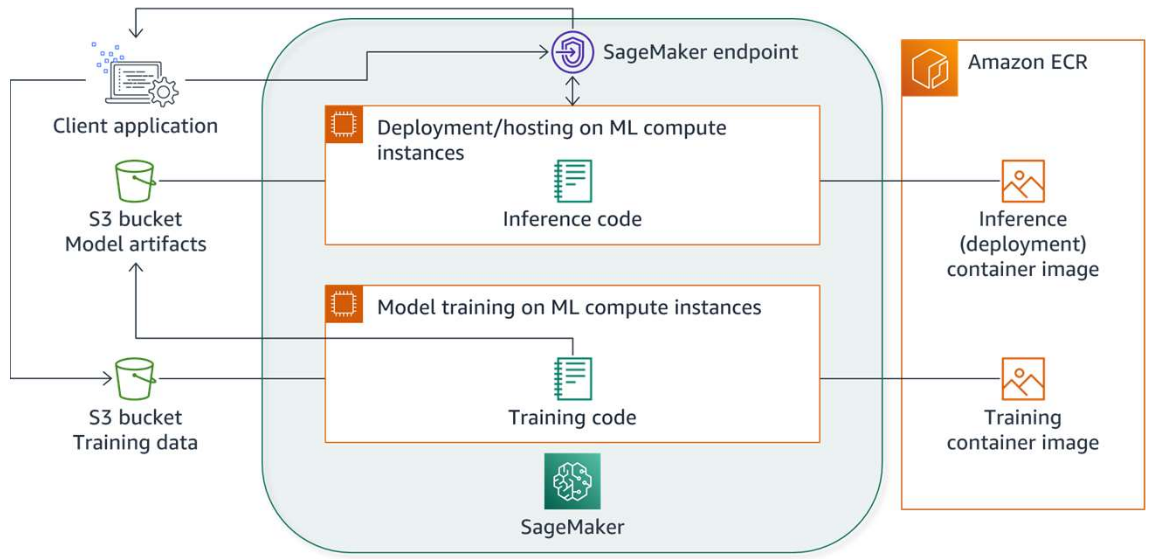
Amazon SageMaker Processing is a tool to analyze data and evaluate ML models. With Processing, you can use a simplified, managed experience on SageMaker to run your data processing steps, such as feature engineering, data validation, and model evaluation. You can choose to put your code on built-in data processing containers or bring your own containers and submit custom jobs to run on the SageMaker managed infrastructure.

SageMaker comes with a suite of built-in algorithm choices as well as pre-trained models and pre-built solution templates to help data scientists start to train and deploy models quickly.

For deploying your ML models, SageMaker provides the option to deploy using a hosted endpoint that is suitable for real-time inference where you want to make one prediction at a time. Alternatively, you can use batch transform to run a set of predictions against an entire dataset.



Training and deploying a model with SageMaker



|Slide number 73
|Instructor notes
|
|Student notes

To train a model in SageMaker, you create a training job. The diagram on this slide shows the AWS services that are used to run a training job and deploy it to production. SageMaker Hosting hosts an endpoint where applications can access the model and have it predict an outcome when given an individual request. This architecture will be used in the demo that is titled "Preparing Data and Training a Model with SageMaker."

The training job includes the following information:

- The URL of the S3 bucket where you stored the training data.
- The compute resources that you want SageMaker to use for model training. Compute resources are ML compute instances that SageMaker manages.
- The URL of the S3 bucket where you want to store the output of the job.
- The Amazon Elastic Container Registry (Amazon ECR) path of where the training code is stored.

After you create the training job, SageMaker launches the ML compute instances and uses the training code and the training dataset to train the model. The service saves the resulting model artifacts and other output in the S3 bucket that you specified for that purpose. SageMaker gets the training image for your selected algorithm from a container that is available through Amazon ECR, which is a repository to store and access containers.

The ML compute instances and the SageMaker endpoint are directly managed within SageMaker. SageMaker is integrated with Amazon S3 and Amazon ECR to simplify using those resources for training and deployment.

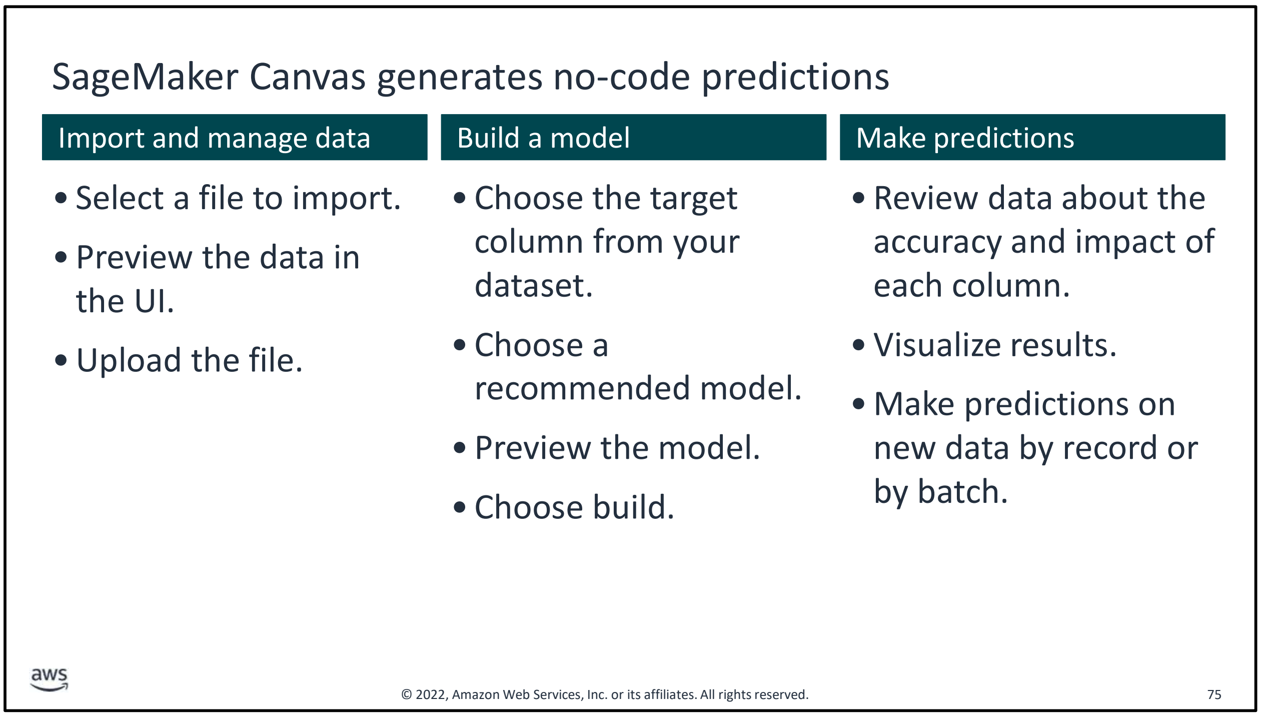
Demo: Preparing Data and Training a Model with SageMaker

- This demo illustrates the steps to create the architecture that was presented earlier in this section and to deploy and test the fraud detection model.
- The recorded demo is included in your online course.

|Slide number 74
|Instructor notes
|This recorded demo cannot be done live because the permissions that are required to perform some steps cannot be granted in the lab environments.
|
|Student notes
This demo illustrates the training and deployment of an ML model by using SageMaker. The demo mirrors the diagram from earlier in this section and uses the insurance claim fraud data that was discussed to generate predictions about whether a claim is fraudulent.

The demo illustrates the following steps:
- Using the AWS SDK for Python (Boto3) to create S3 buckets.
- Using the Pandas library to stage, clean, and prepare training and validation datasets.
- Obtaining the Docker image for the linear-learner algorithm by using the Boto3 API and SageMaker API. The linear-learner algorithms is included with SageMaker, and it is a good fit for this binary classification problem.
- Starting the training process by using the training dataset and an EC2 instance that is specifically designed for ML model training and managed by SageMaker.
- Configuring a SageMaker Hosting endpoint and deploying the trained model to the endpoint.

- Testing the deployed endpoint and assessing the precision, recall, and F1 score of the model.



SageMaker Canvas generates no-code predictions

Import and manage data	Build a model	Make predictions
• Select a file to import. • Preview the data in the UI. • Upload the file.	• Choose the target column from your dataset. • Choose a recommended model. • Preview the model. • Choose build.	• Review data about the accuracy and impact of each column. • Visualize results. • Make predictions on new data by record or by batch.

|Slide number 75
|Instructor notes
|
|Student notes

SageMaker has been designed to abstract a lot of the infrastructure and environment management work that is required to support ML workloads. By using the service, data engineers and data scientists can focus on the goals and outcomes. SageMaker Canvas takes the abstraction to another level and provides an interface where a business analyst who isn't a data scientist can create models and get predictions.

Demo: Preparing Data and Training a Model with SageMaker Canvas

- This demo illustrates how to use SageMaker Canvas to create an ML model without the need to write or use code.
- The recorded demo is included in your online course.

|Slide number 76

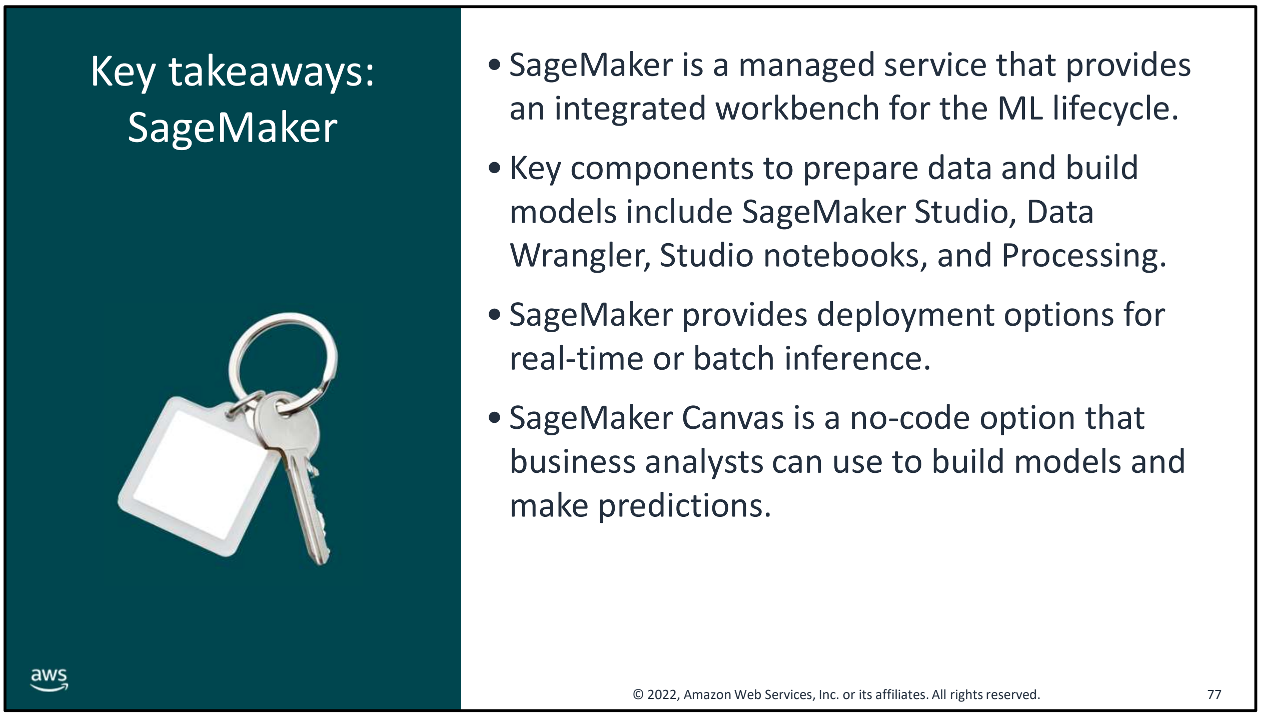
|Instructor notes

|This recorded demo cannot be done live because the permissions that are required to perform some steps cannot be granted in the lab environments.

|

|Student notes

This demo uses the same dataset and business problem as the Preparing Data and Training a Model with SageMaker demo. This demo highlights how users without ML experience could use the data to train a model without using any notebooks or writing any code.



Key takeaways: SageMaker

- SageMaker is a managed service that provides an integrated workbench for the ML lifecycle.
- Key components to prepare data and build models include SageMaker Studio, Data Wrangler, Studio notebooks, and Processing.
- SageMaker provides deployment options for real-time or batch inference.
- SageMaker Canvas is a no-code option that business analysts can use to build models and make predictions.

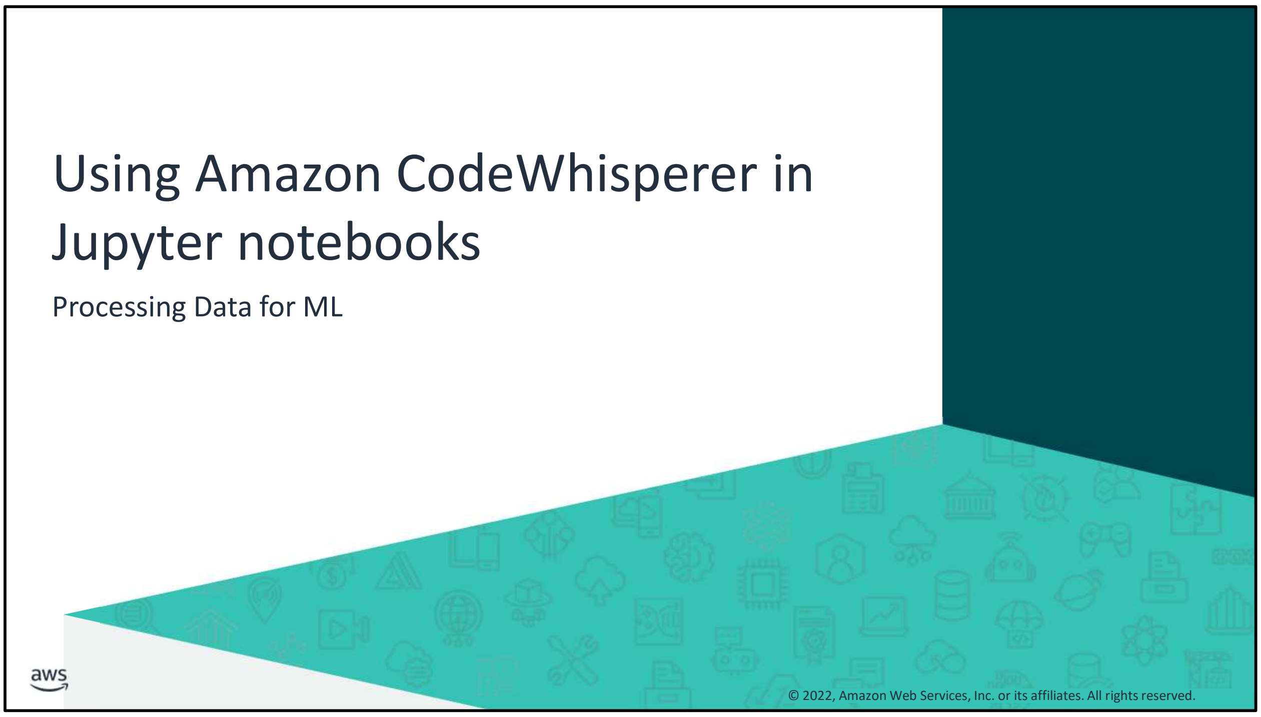
|Slide number 77
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

SageMaker is a managed service with tools and features that work across the ML lifecycle.

SageMaker Studio provides an integrated IDE and provides access to other SageMaker features in one place.

With SageMaker Hosting, you can deploy real-time inferencing or use batch transform to process inferencing on a whole dataset as a batch.

SageMaker Canvas gives business analysts the power of ML without having to code at all.



Using Amazon CodeWhisperer in Jupyter notebooks

Processing Data for ML

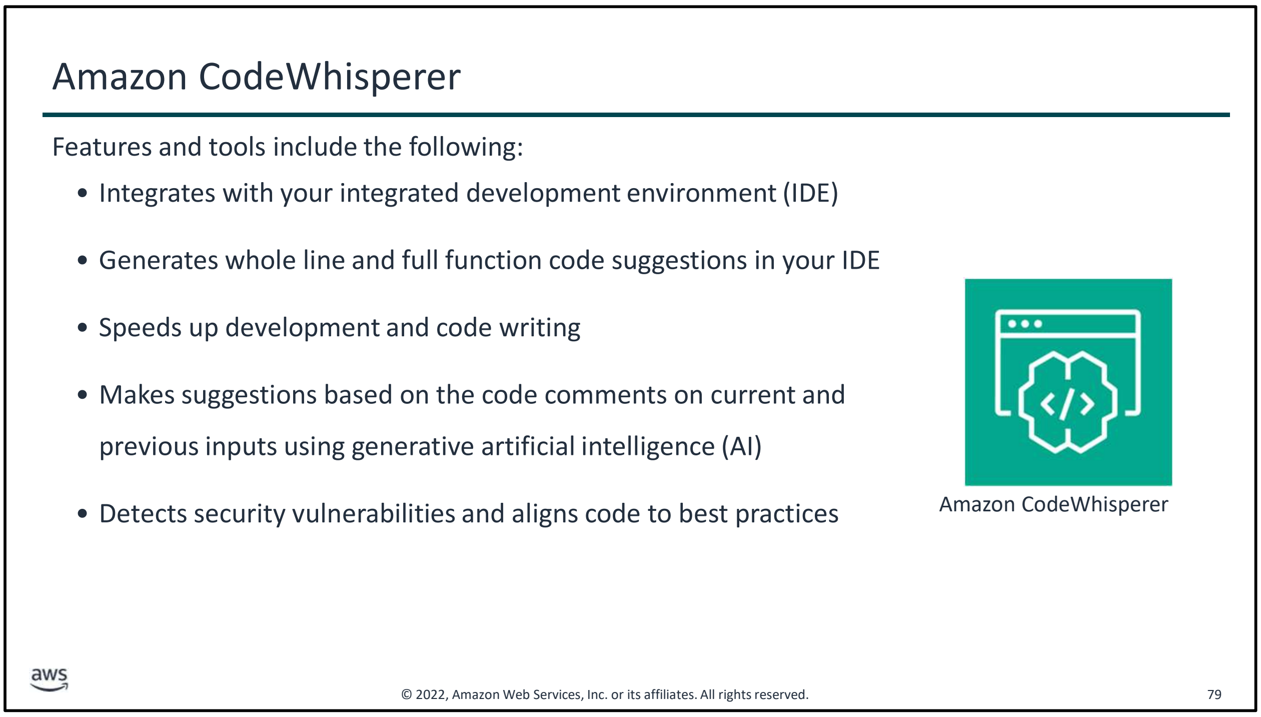
|Slide number 78

|Instructor notes

|Amazon CodeWhisperer can help developers write code in ML applications faster.

|Student notes

In this section, you learn how Amazon CodeWhisperer can help you write code in your ML application faster.



Amazon CodeWhisperer

Features and tools include the following:

- Integrates with your integrated development environment (IDE)
- Generates whole line and full function code suggestions in your IDE
- Speeds up development and code writing
- Makes suggestions based on the code comments on current and previous inputs using generative artificial intelligence (AI)
- Detects security vulnerabilities and aligns code to best practices

Amazon CodeWhisperer

|Slide number 79
|Instructor notes
|Amazon CodeWhisperer can help software development engineers (SDEs) write code in machine learning (ML) applications faster.
|Student notes
Amazon CodeWhisperer is an AI coding companion that generates whole line and full function code suggestions in your IDE. Using Amazon CodeWhisperer enables machine learning (ML) engineers to speed up development and code writing. Amazon CodeWhisperer makes suggestions based on the code comments on current and previous inputs using generative AI. CodeWhisperer is best used for undifferentiated, repetitive, common coding tasks.

Aside from helping to generate code, Amazon CodeWhisperer also scans your code to detect hard-to-find security vulnerabilities and provides code suggestions to remediate them. It aligns code to best practices for correcting security vulnerabilities outlined by Open Worldwide Application Security Project (OWASP) and other best practices.

For more information about the languages and platforms supported by Amazon CodeWhisperer, see the resources for this module on the Content Resources page of your course.

Amazon CodeWhisperer and generative AI

Amazon CodeWhisperer uses the following:

- Generative AI technology
- Pre-trained models called foundation models (FMs)
  - Large language models (LLMs)
- Prompt engineering

|Slide number 80
|Instructor notes
|This is a brief overview of foundation models and large language models.
|Student notes

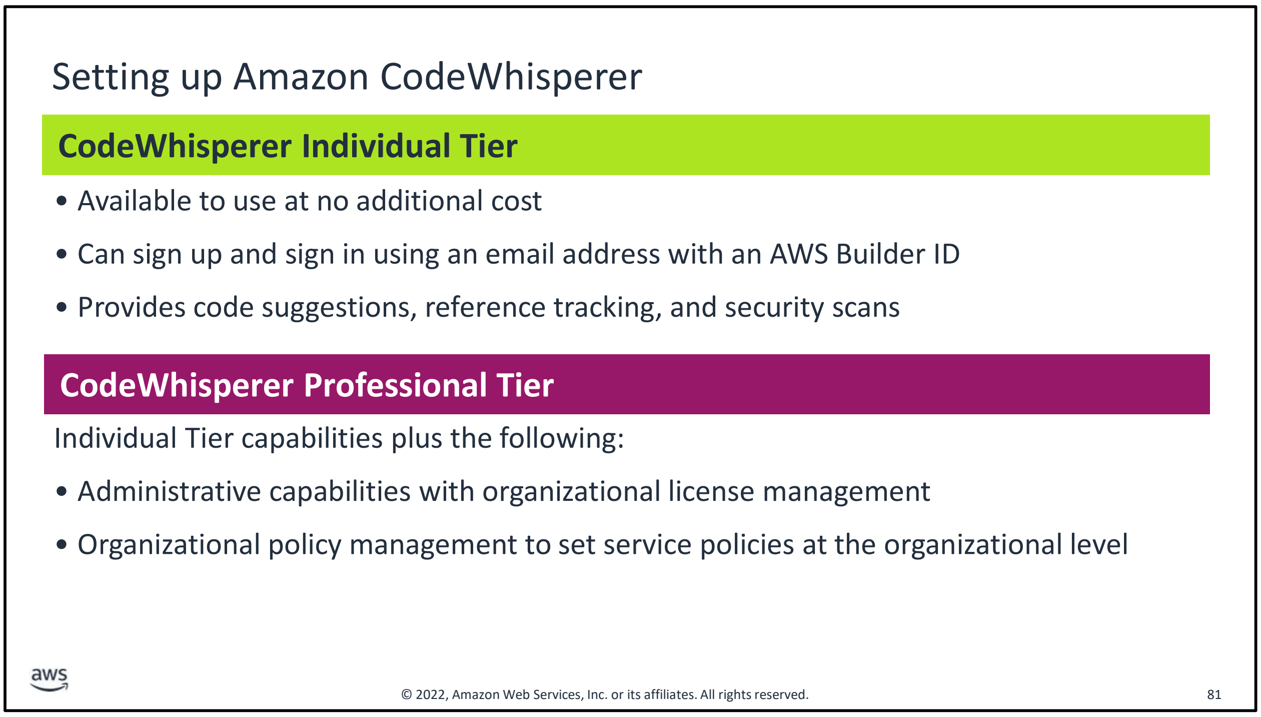
Amazon CodeWhisperer uses generative AI. Generative AI can create new content and ideas powered by large pre-trained models commonly referred to as **foundation models (FMs)**.

A subset of FMs, called **large language models** (LLMs), are trained on trillions of words across many natural language tasks.

These LLMs can understand, learn, and generate text that's nearly indistinguishable from text produced by humans.

With traditional ML models, in order to achieve each specific task, customers need to gather labeled data, train a model, and deploy that model. With LLMs, instead of gathering labeled data for each model and training multiple models, customers can use the same pretrained LLM to adapt various tasks such as text generation, summarization, or sentiment analysis. LLMs can also be customized to perform domain-specific functions that are differentiating to their businesses, using only a small fraction of the data and compute required to train a model from scratch.

While working with CodeWhisperer, you interact with the FMs using a text input or prompt in the form of code comments. Finding the right prompt so that the model generates the expected output is called **prompt engineering**.



Setting up Amazon CodeWhisperer

CodeWhisperer Individual Tier

- Available to use at no additional cost
- Can sign up and sign in using an email address with an AWS Builder ID
- Provides code suggestions, reference tracking, and security scans

CodeWhisperer Professional Tier

Individual Tier capabilities plus the following:

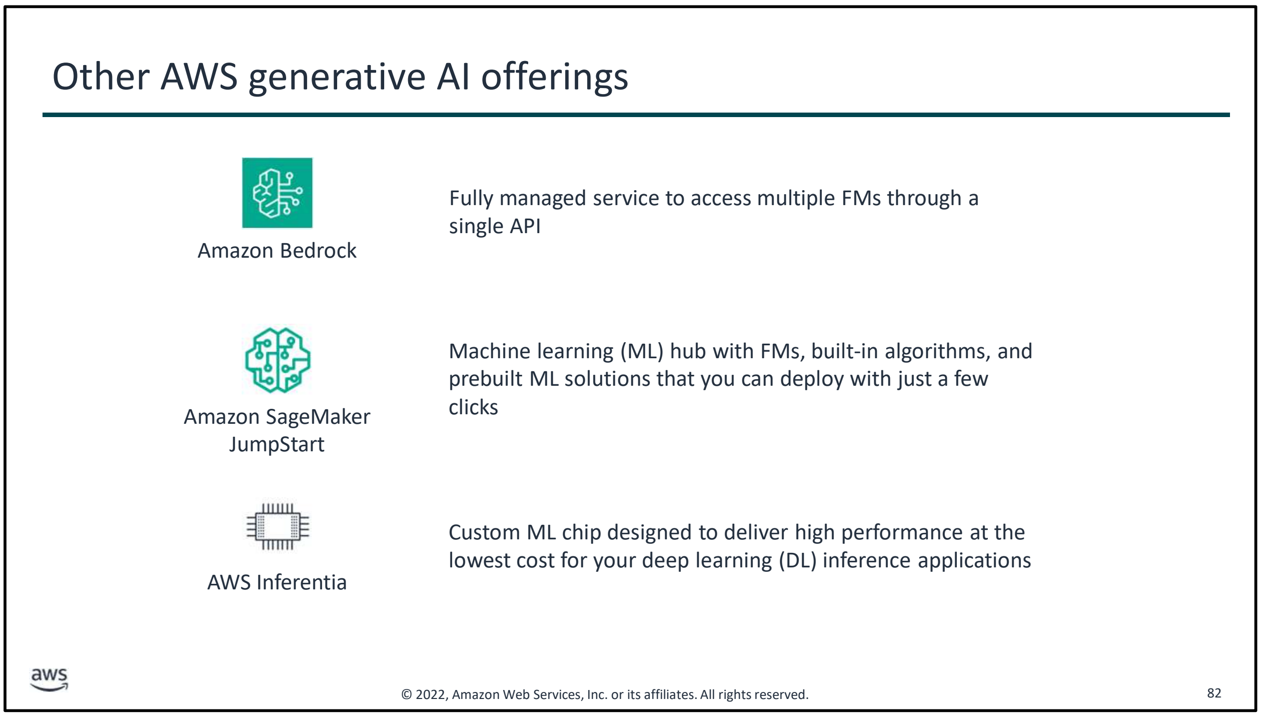
- Administrative capabilities with organizational license management
- Organizational policy management to set service policies at the organizational level

|Slide number 81
|Instructor notes
|
|Student notes
There are two tiers of service for Amazon CodeWhisperer.

CodeWhisperer Individual Tier
The CodeWhisperer Individual Tier is available to use at no additional cost. Individual developers can sign up and sign in using an email address with an AWS Builder ID to start using CodeWhisperer within minutes. The Individual Tier provides code suggestions, reference tracking, and security scans.

CodeWhisperer Professional Tier
In addition to the capabilities offered in the Individual Tier, the CodeWhisperer Professional Tier offers administrative capabilities to organizations that want to provide their developers with access to CodeWhisperer. Administrators get organizational license management to centrally manage which developers in the organization should have access to CodeWhisperer. They also get organizational policy management to set service policies at the organizational level, such as whether developers are allowed to receive code suggestions that might be similar to particular open-source training data.

Choosing a CodeWhisperer tier depends on the use case, the IDE, and platform that you’re developing on. For more information about how to set up Amazon CodeWhisperer, see “Setting up Amazon CodeWhisperer” in the Content Resources page of your course.



Other AWS generative AI offerings

Amazon Bedrock — Fully managed service to access multiple FMs through a single API

Amazon SageMaker JumpStart — Machine learning (ML) hub with FMs, built-in algorithms, and prebuilt ML solutions that you can deploy with just a few clicks

AWS Inferentia — Custom ML chip designed to deliver high performance at the lowest cost for your deep learning (DL) inference applications

~developer notes: Add https://aws.amazon.com/generative-ai/ to content resources

|Slide number 82
|Instructor notes
|Student notes
AWS offers other generative AI services. Some of them include the following:

- Amazon Bedrock: A fully managed service that makes FMs from Amazon and leading AI startups available through an API. This means that you can choose from various FMs to find the model that's best suited for your use case. Amazon Bedrock helps developers create generative AI applications that can deliver up-to-date answers based on proprietary knowledge sources.

- Amazon SageMaker JumpStart: Amazon SageMaker JumpStart helps you quickly get started with ML. It provides a set of solutions for the most common use cases that can be deployed readily in just a few steps. The solutions are fully customizable and showcase the use of AWS CloudFormation templates and reference architectures so you can accelerate your ML journey.

- AWS Inferentia: AWS Inferentia is a custom ML chip designed by AWS that you can use for high-performance inference predictions. In order to use the chip, set up an Amazon

Elastic Compute Cloud (Amazon EC2) instance and use the AWS Neuron SDK to invoke the Inferentia chip.

For more information on other generative AI offerings at AWS, see the Content Resources section of your course.

Demo: Creating a Linear Regression Model Using Amazon CodeWhisperer

- This demonstration illustrates how Amazon CodeWhisperer can accelerate coding tasks in ML applications.
- The recorded demo is included in your online course.

|Slide number 83

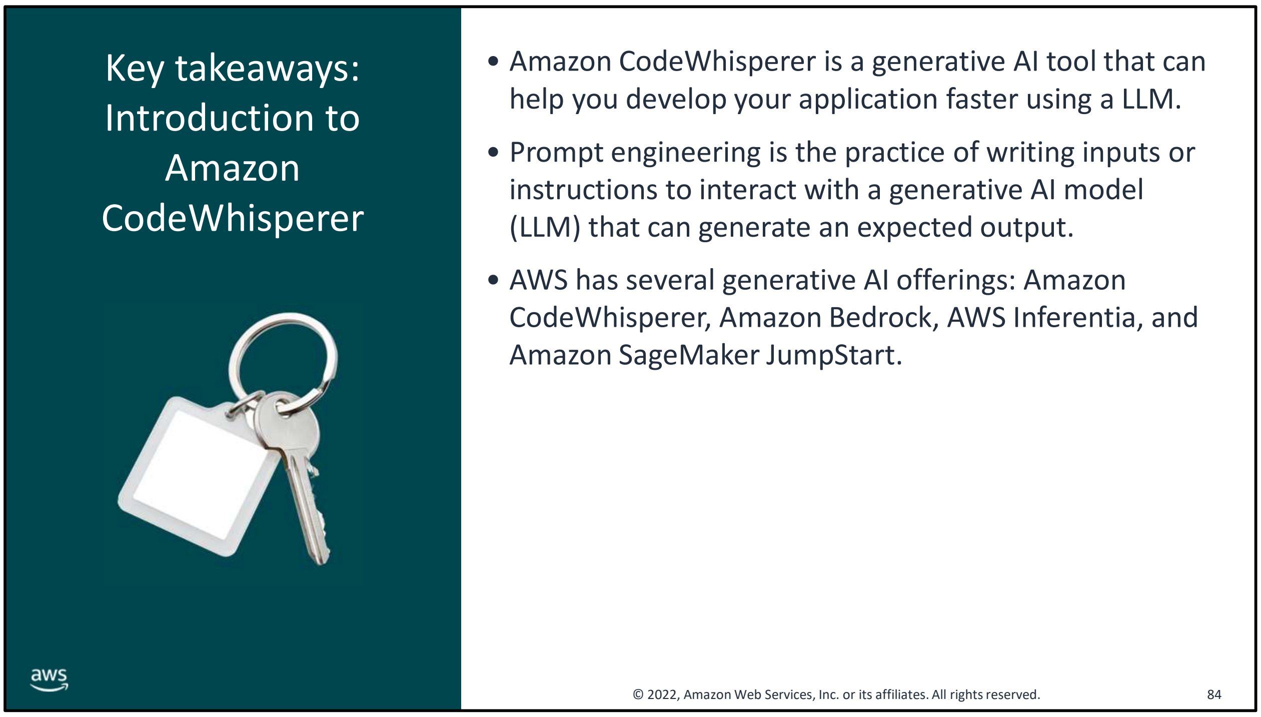
|Instructor notes

|This recorded demo cannot be done live because the permissions that are required to perform some steps can’t be granted in the lab environments.

|

|Student notes

This demo highlights how CodeWhisperer increases developer productivity by suggesting code comments and suggestions in Jupyter notebooks.



Key takeaways: Introduction to Amazon CodeWhisperer

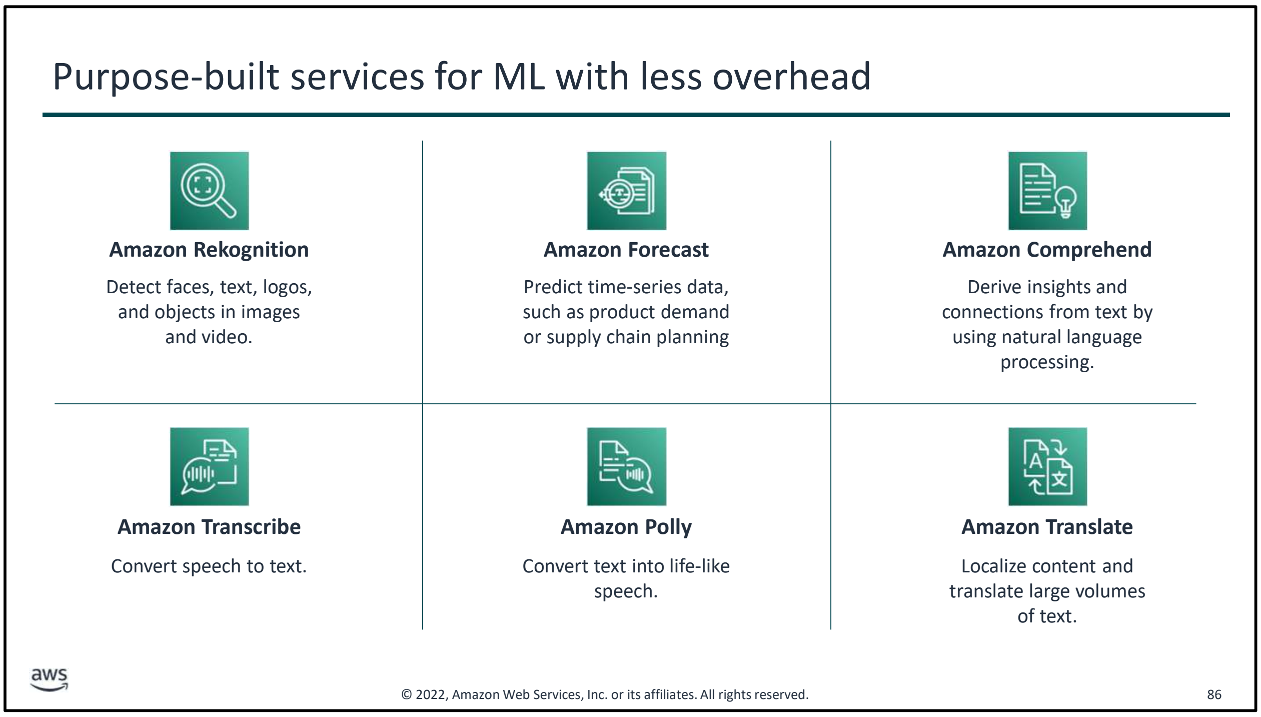
- Amazon CodeWhisperer is a generative AI tool that can help you develop your application faster using a LLM.
- Prompt engineering is the practice of writing inputs or instructions to interact with a generative AI model (LLM) that can generate an expected output.
- AWS has several generative AI offerings: Amazon CodeWhisperer, Amazon Bedrock, AWS Inferentia, and Amazon SageMaker JumpStart.

|Slide number 84
|Instructor notes
|
|Student notes
Using Amazon CodeWhisperer can make developing code faster and more secure. The more precise your prompts are, the better Amazon CodeWhisperer responds. AWS has several generative AI offerings.

AI/ML services on AWS

Processing Data for ML

|Slide number 85
|Instructor notes
|
|Student notes
This section introduces additional AWS services that provide AI/ML functionality for common use cases.



Purpose-built services for ML with less overhead

Amazon Rekognition
Detect faces, text, logos, and objects in images and video.

Amazon Forecast
Predict time-series data, such as product demand or supply chain planning

Amazon Comprehend
Derive insights and connections from text by using natural language processing.

Amazon Transcribe
Convert speech to text.

Amazon Polly
Convert text into life-like speech.

Amazon Translate
Localize content and translate large volumes of text.

~https://aws.amazon.com/machine-learning/ai-services/

|Slide number 86

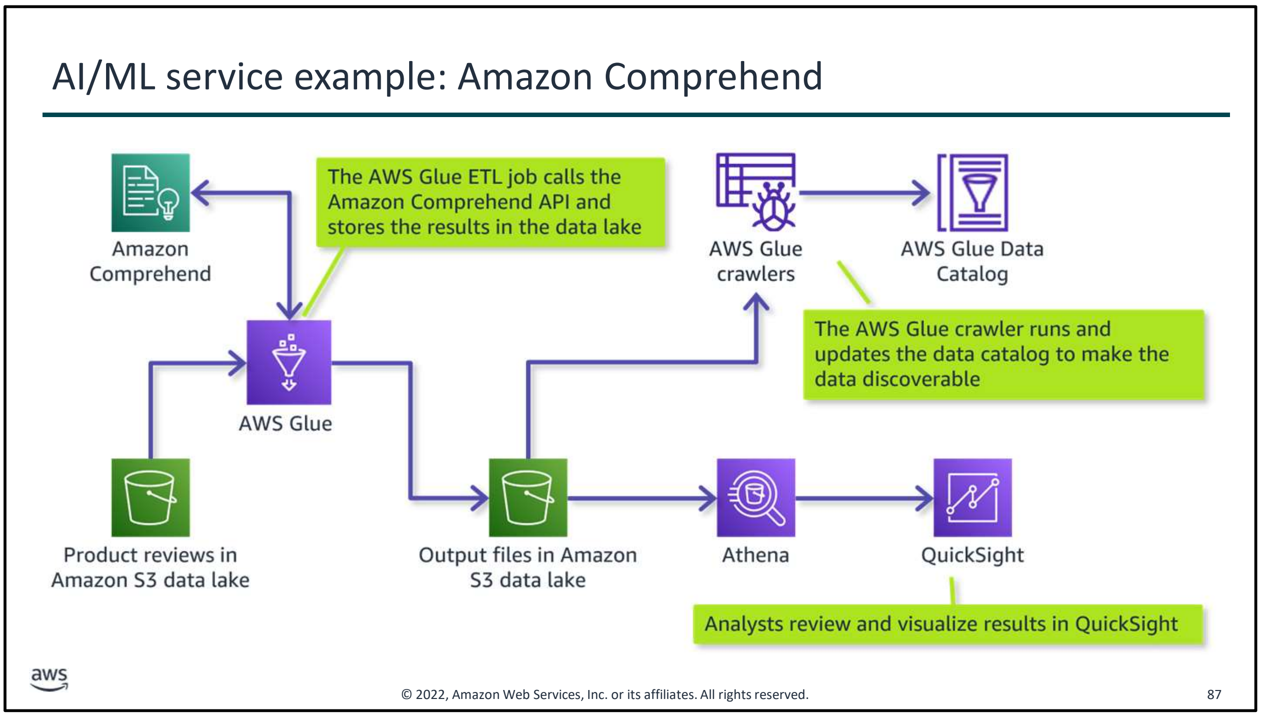
|Instructor notes

|The AI Services page within the Machine Learning on AWS section of the aws.amazon.com website provides a full set of available AI features. You might choose to highlight a different set based on students' experiences or fields of study. These services are presented to highlight the many places that AI/ML is being used for tasks that have become much more common.

|

|Student notes

The AWS services that are described on this slide highlight just a few of the AWS options to use AI/ML to perform specific tasks. Other cloud providers have similar offerings, and you probably run into them in your daily life often without even thinking about it. You can use many of these services with little ML knowledge and can integrate them into your applications. The resources page in your course provides a link to the AWS AI services page, where you will find a complete list of AWS AI services with links to go deeper on any of them.



AI/ML service example: Amazon Comprehend

Amazon Comprehend
The AWS Glue ETL job calls the Amazon Comprehend API and stores the results in the data lake
AWS Glue crawlers
AWS Glue Data Catalog
The AWS Glue crawler runs and updates the data catalog to make the data discoverable
AWS Glue
Product reviews in Amazon S3 data lake
Output files in Amazon S3 data lake
Athena
QuickSight
Analysts review and visualize results in QuickSight

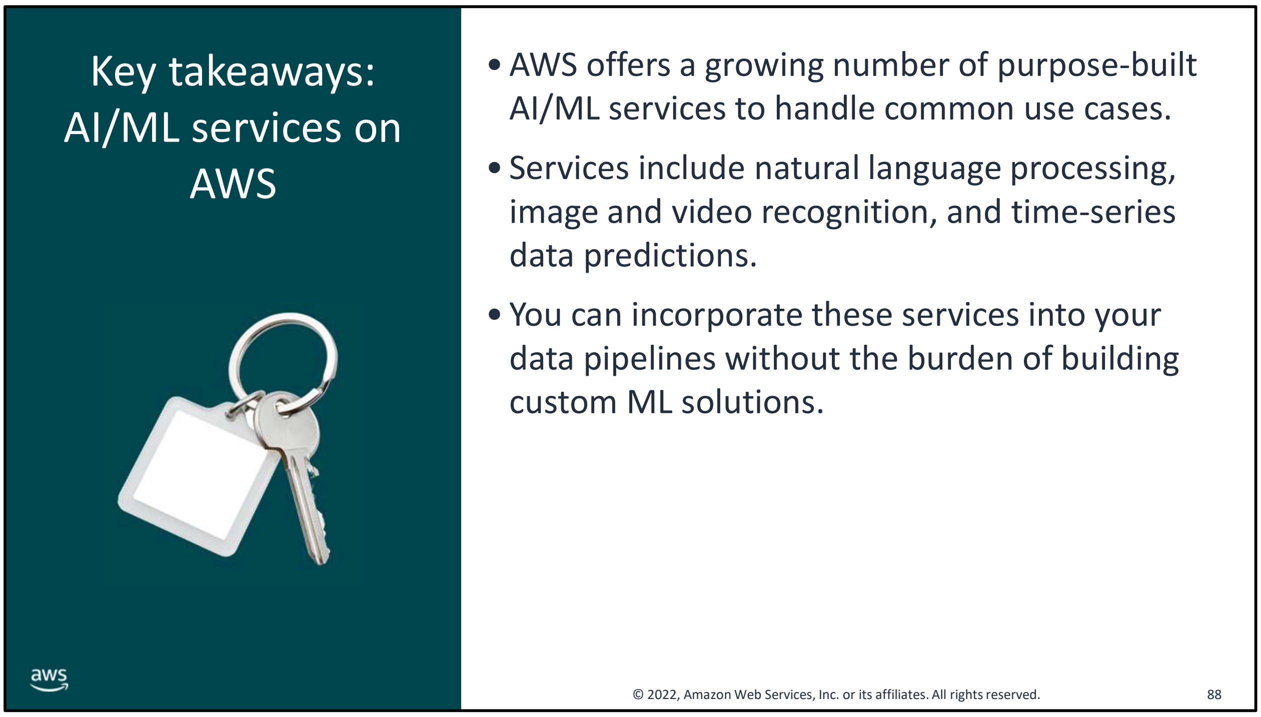
|Slide number 87
|Instructor notes
|You might choose a different service to highlight based on interests or relevance to the audience. Each service page provides an overview of how the service works and provides use case examples and AWS customer examples. You might also search the AWS Blog section of the aws.amazon.com website for the service of interest to see specific examples and architectures for using these services.
|
|Student notes
Amazon Comprehend is one example of a service that simplifies a specific type of ML problem, specifically natural language processing. Amazon Comprehend can analyze a document or set of documents to gather insights about it. The service uses a pre-trained model that is continuously trained on a large body of text, so you don't need to provide your own training data.

Some insights that Amazon Comprehend develops about a document include the following items:
- A list of entities, such as people, places, and locations, that are identified in a document.
- Key phrases that appear in a document. For example, a document about a basketball game might return the names of the teams, the name of the venue, and the final score.

- The dominant sentiment of a document. Sentiment can be positive, neutral, negative, or mixed.

You can also create custom models if needed.

The example on the slide illustrates one way that you might use Amazon Comprehend to process data that has been ingested into your pipeline. In this example, customer product reviews have been collected in a data lake. An AWS Glue ETL job gets the review data and calls the Amazon Comprehend API to process each review. The AWS Glue job writes the results of each record to a separate location in the data lake. An AWS Glue crawler runs to discover the output data and add its metadata to the AWS Glue Data Catalog. Data analysts can log in to Amazon QuickSight and create visualizations of the data that are backed by the Amazon S3 query capabilities in Amazon Athena. The Analyzing and Visualizing Data module talks in greater detail about Athena and QuickSight.



Key takeaways: AI/ML services on AWS

- AWS offers a growing number of purpose-built AI/ML services to handle common use cases.
- Services include natural language processing, image and video recognition, and time-series data predictions.
- You can incorporate these services into your data pipelines without the burden of building custom ML solutions.

|Slide number 88
|Instructor notes
|
|Student notes

Here are a few key points to summarize this section.

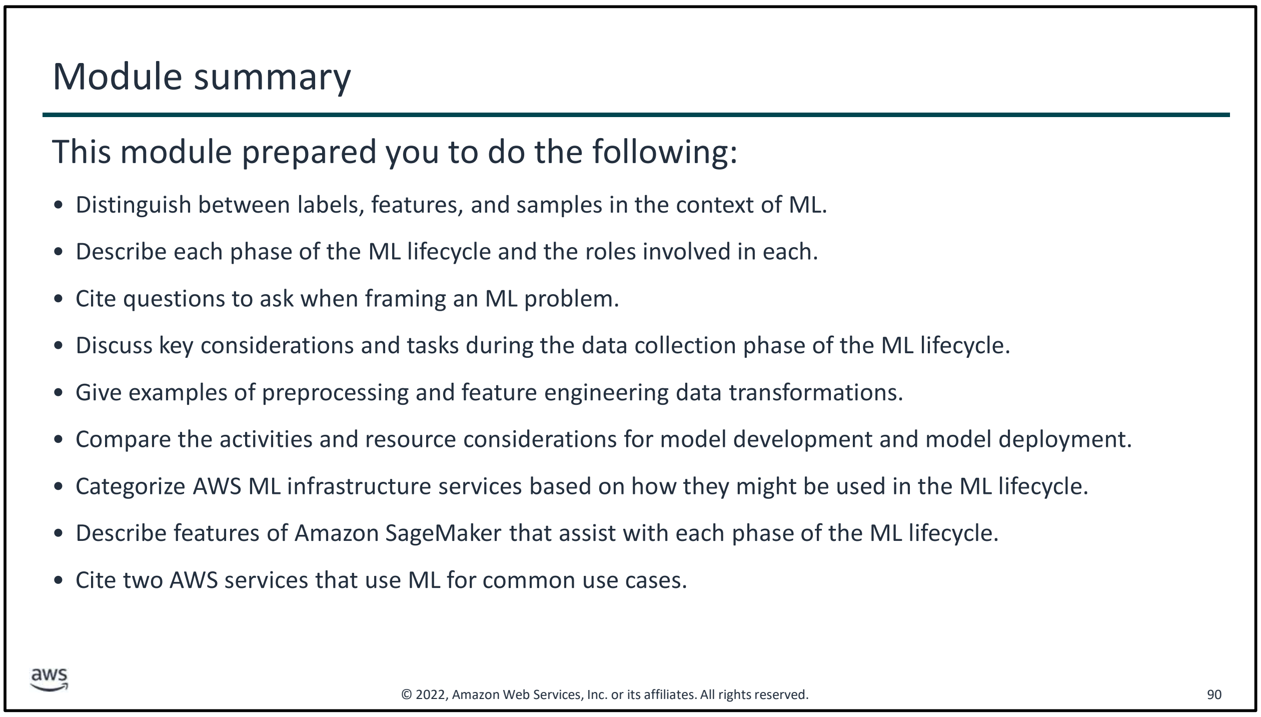
AWS offers purpose-built services to address common ML processing use cases, such as natural language processing, image recognition, and time-series data.

You can incorporate these services into the ingestion or processing layers of your pipeline with much less overhead than building your own ML solution.

Module wrap-up

Processing Data for ML

|Slide number 89
|Instructor notes
|
|Student notes
This section summarizes what you have learned and brings the module to a close.

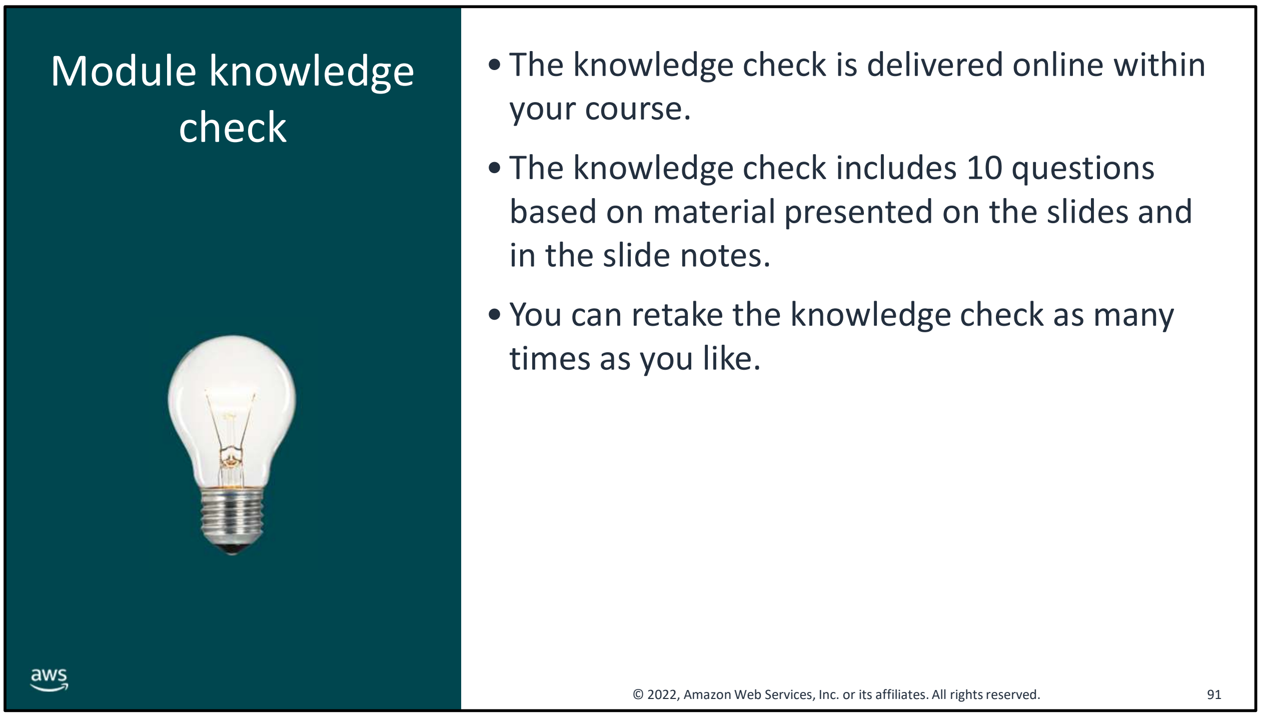


Module summary

This module prepared you to do the following:

- Distinguish between labels, features, and samples in the context of ML.
- Describe each phase of the ML lifecycle and the roles involved in each.
- Cite questions to ask when framing an ML problem.
- Discuss key considerations and tasks during the data collection phase of the ML lifecycle.
- Give examples of preprocessing and feature engineering data transformations.
- Compare the activities and resource considerations for model development and model deployment.
- Categorize AWS ML infrastructure services based on how they might be used in the ML lifecycle.
- Describe features of Amazon SageMaker that assist with each phase of the ML lifecycle.
- Cite two AWS services that use ML for common use cases.

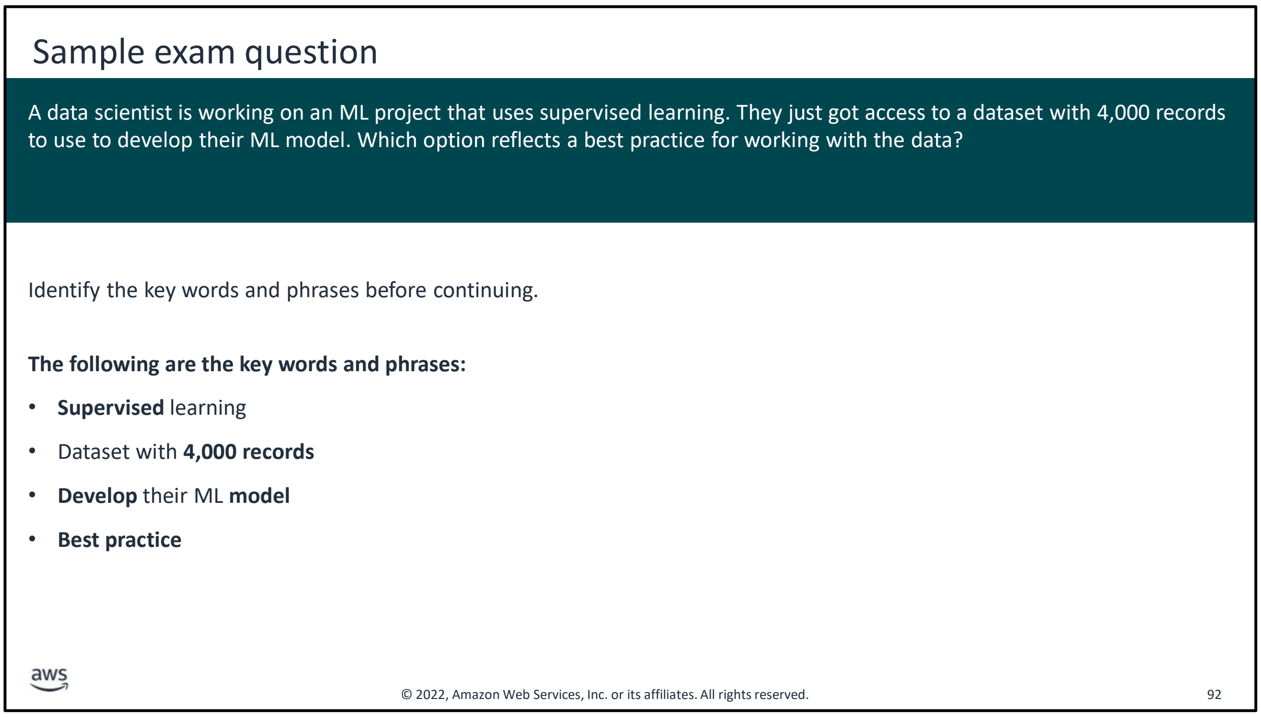
|Slide number 90
|Instructor notes
|
|Student notes
This modules provided a brief look at machine learning and focused on the ML lifecycle, which guides how data engineers and data scientists collect and process data for ML models. The module took a quick look at general ML concepts and described the activities that are performed in each phase of the ML lifecycle. You learned about framing the ML problem, collecting and preprocessing data, feature engineering, model development, and model deployment. You also looked at the AWS infrastructure services that you can use to implement the ML lifecycle. Finally, you were introduced to a few purpose-built AI/ML services that you can use to solve common use cases without needing a lot of ML knowledge.



Module knowledge check

- The knowledge check is delivered online within your course.
- The knowledge check includes 10 questions based on material presented on the slides and in the slide notes.
- You can retake the knowledge check as many times as you like.

|Slide number 91
|Instructor notes
|
|Student notes
Use your online course to access the knowledge check for this module.



Sample exam question

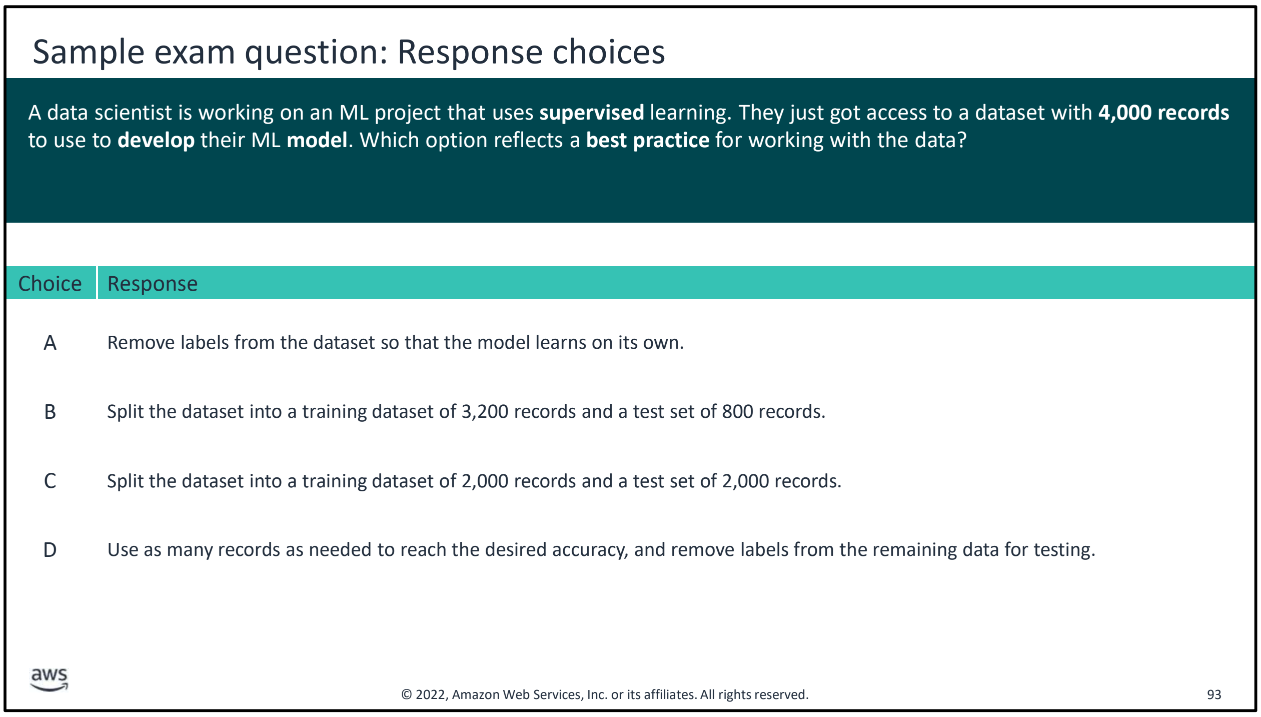
A data scientist is working on an ML project that uses supervised learning. They just got access to a dataset with 4,000 records to use to develop their ML model. Which option reflects a best practice for working with the data?

Identify the key words and phrases before continuing.

The following are the key words and phrases:

- **Supervised** learning
- Dataset with **4,000 records**
- **Develop** their ML **model**
- **Best practice**

|Slide number 92
|Instructor notes
|The key words section is animated to be revealed on click.
|
|Student notes
The question notes that the data scientist will use a 4,000-record dataset to develop an ML model. The question asks for a related best practice.



Sample exam question: Response choices

A data scientist is working on an ML project that uses **supervised** learning. They just got access to a dataset with **4,000 records** to use to **develop** their ML **model**. Which option reflects a **best practice** for working with the data?

Choice	Response
A	Remove labels from the dataset so that the model learns on its own.
B	Split the dataset into a training dataset of 3,200 records and a test set of 800 records.
C	Split the dataset into a training dataset of 2,000 records and a test set of 2,000 records.
D	Use as many records as needed to reach the desired accuracy, and remove labels from the remaining data for testing.

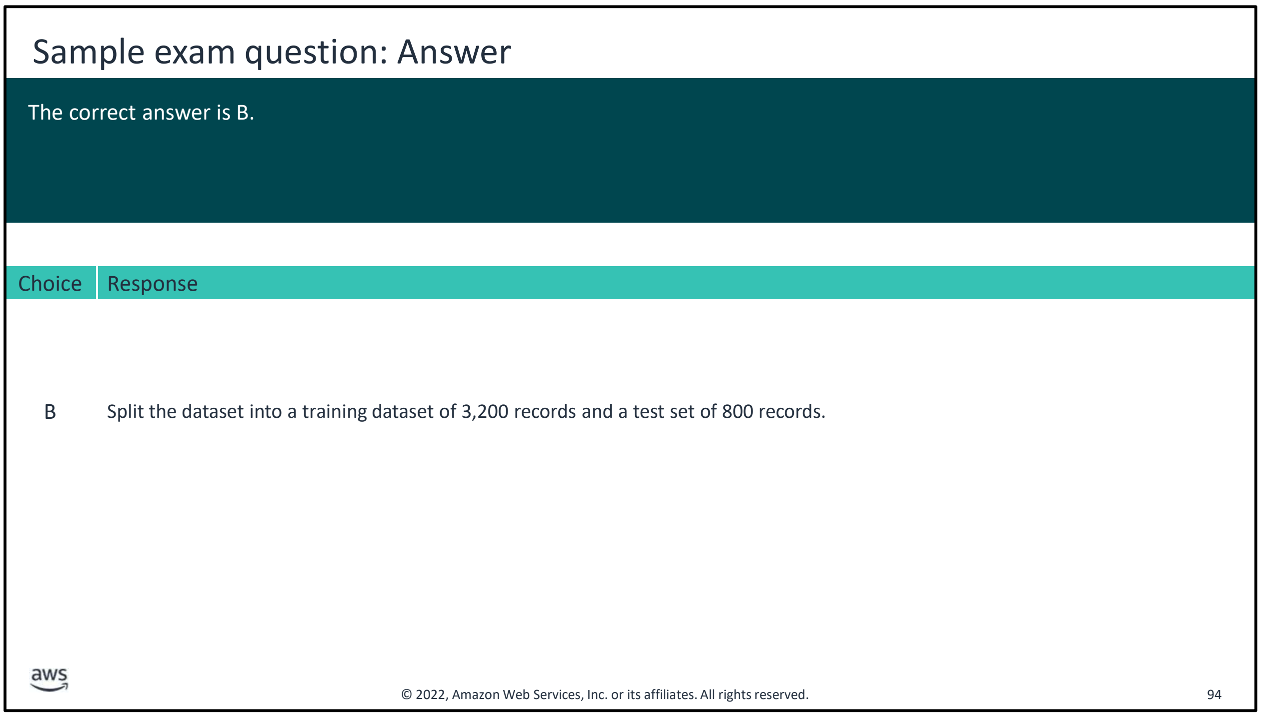
|Slide number 93

|Instructor notes

|

|Student notes

Use the key words that you identified on the previous slide to review each of the responses to determine which one best addresses the question that was presented.


Sample exam question: Answer

The correct answer is B.

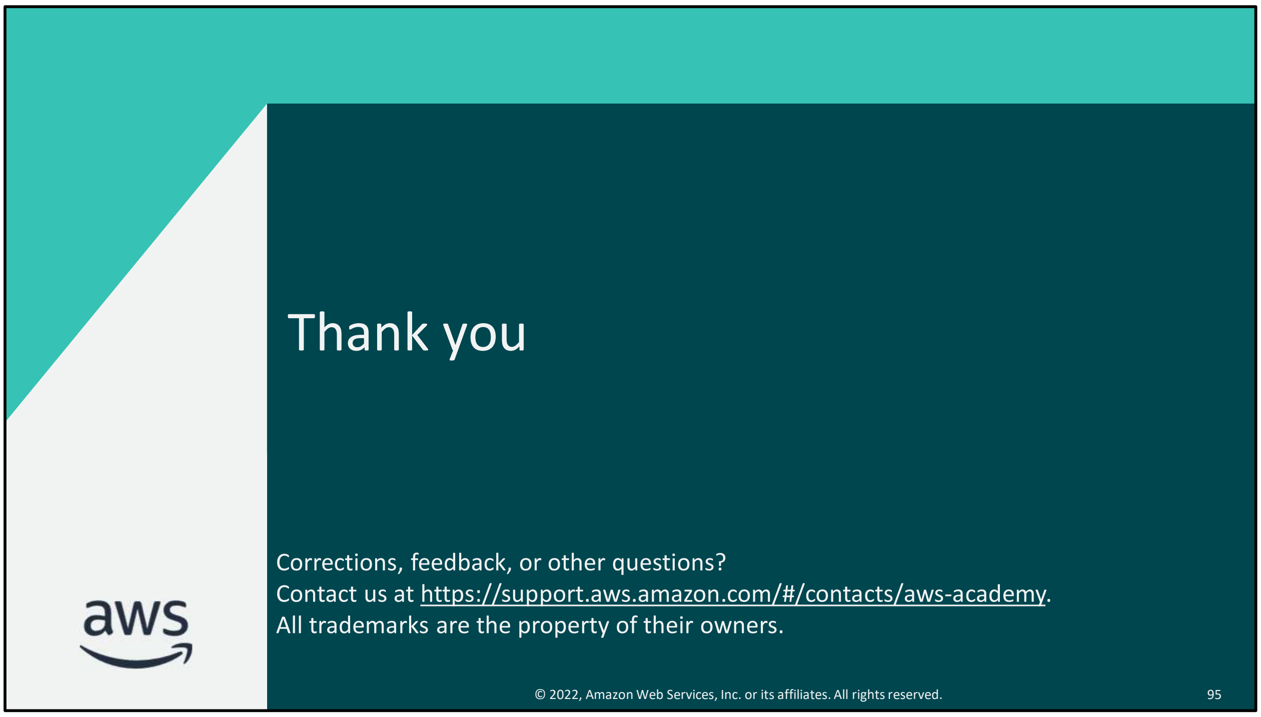
Choice	Response
B	Split the dataset into a training dataset of 3,200 records and a test set of 800 records.

|Slide number 94
|Instructor notes
|
|Student notes
Choice A (Remove labels from the dataset so that the model learns on its own) is incorrect because the question states that this is a supervised learning model.

Choice C (Split the dataset into a training dataset of 2,000 records and a test set of 2,000 records) is not the best choice because it doesn't use enough of the available data to train the dataset.

Choice D (Use as many records as needed to reach the desired accuracy, and remove labels from the remaining data for testing) is not the best answer because you want to split your data to suit the project and set some data aside for testing as a percentage of the total dataset.

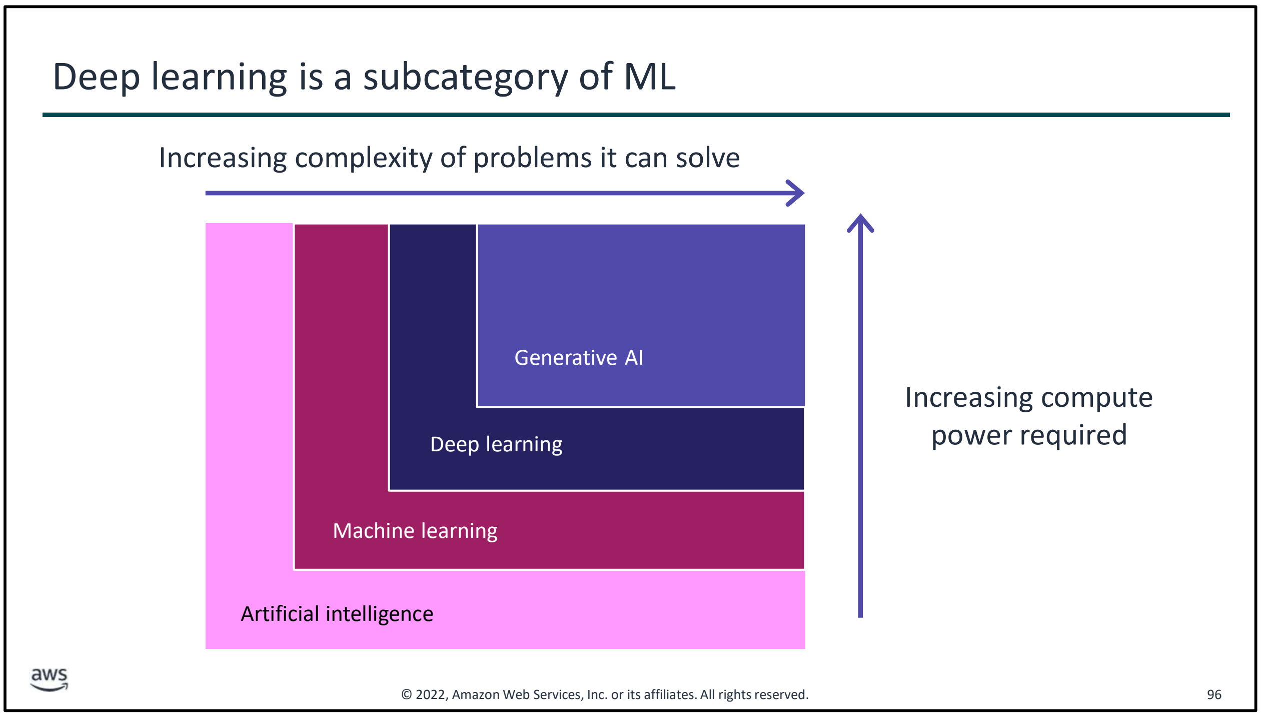
The correct answer is choice B (Split the dataset into a training dataset of 3,200 records and a test set of 800 records). This is in line with the best practice of using 70 to 80 percent of your data for training and the remaining 20 to 30 percent for testing.



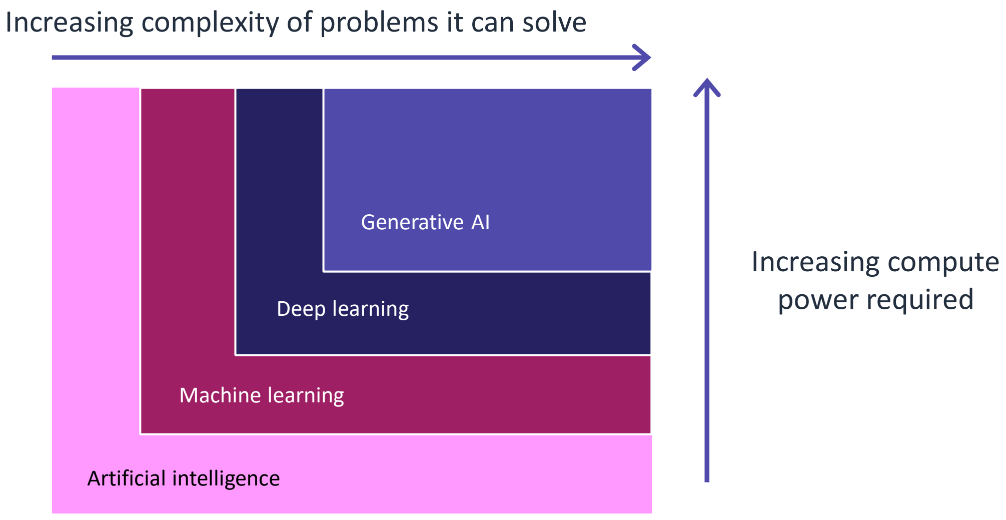
Thank you

Corrections, feedback, or other questions?
Contact us at https://support.aws.amazon.com/#/contacts/aws-academy.
All trademarks are the property of their owners.

~~Script: Thanks for watching!
~
|Slide number 95
|Instructor notes
|
|Student notes
That concludes this module. The Content Resources page of your course includes links to additional resources that are related to this module.



Deep learning is a subcategory of ML

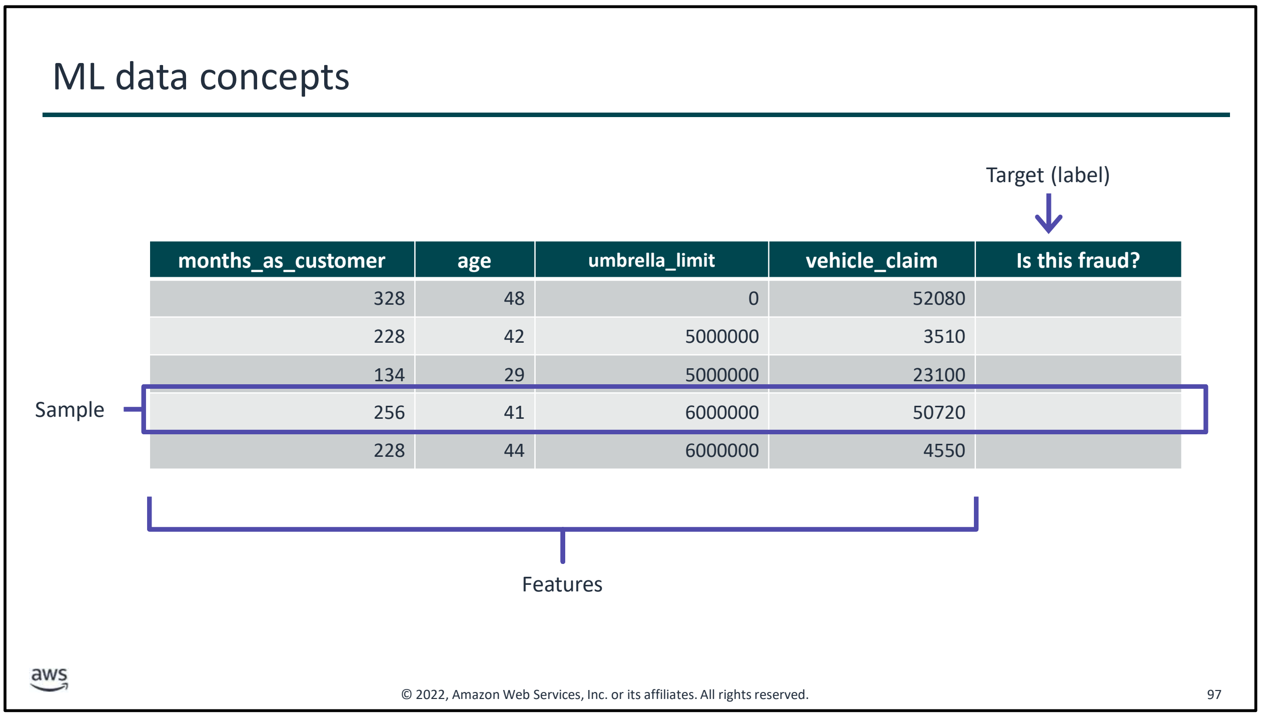

Increasing complexity of problems it can solve
Generative AI
Deep learning
Machine learning
Artificial intelligence
Increasing compute power required

|Slide number 96
|Instructor notes
|
|Student notes

The Data-Driven Organizations module introduces ML as a subcategory of AI. Deep learning is a subcategory of ML whose algorithms use neural networks. The Data-Driven Organizations module introduces ML as a subcategory of AI. Deep learning is a subcategory of ML whose algorithms use neural networks. Generative AI is a subcategory of deep learning.

A *neural network* is based on a collection of connected units or nodes that loosely models the neurons in a biological brain. With deep learning, a model can recognize patterns that are several magnitudes more complex than other ML models could. A deep learning example is a machine that can learn to play chess by playing against itself.

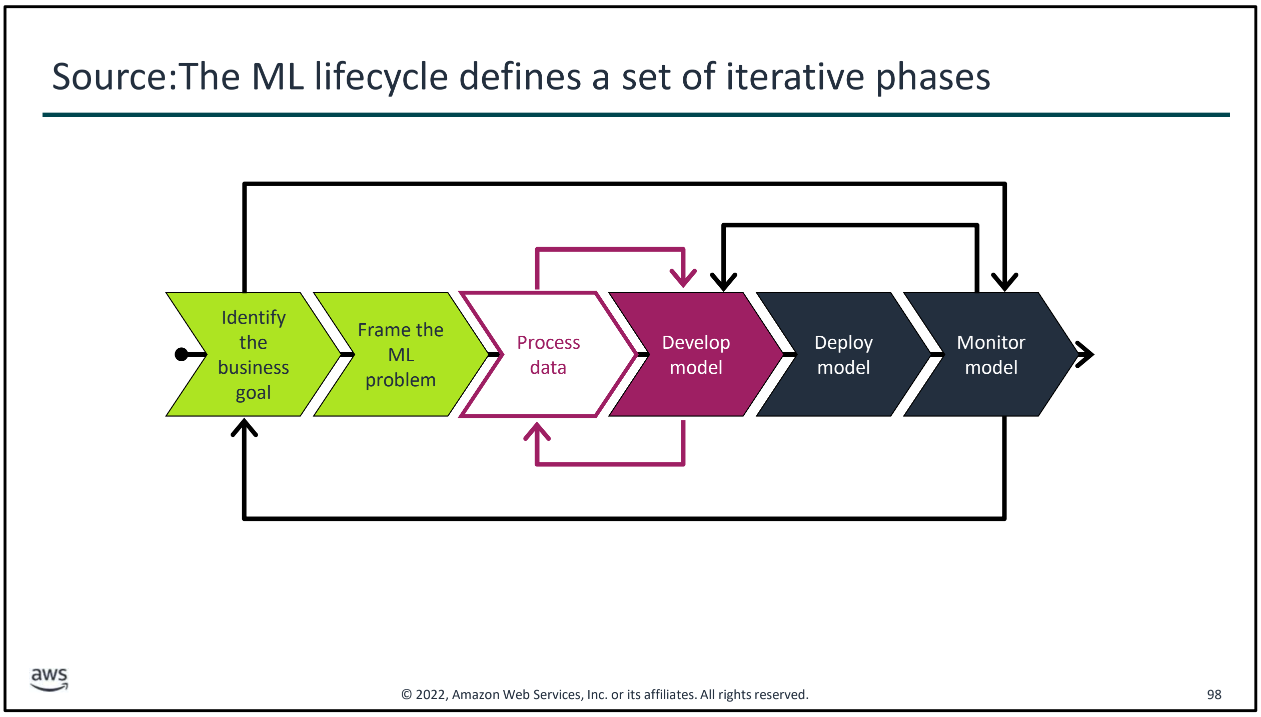
Deep learning can solve more complex problems but requires more compute power than other methods of ML. The idea of deep learning and neural networks has been around for decades, but the computing power that is needed to support it has only recently been readily available.



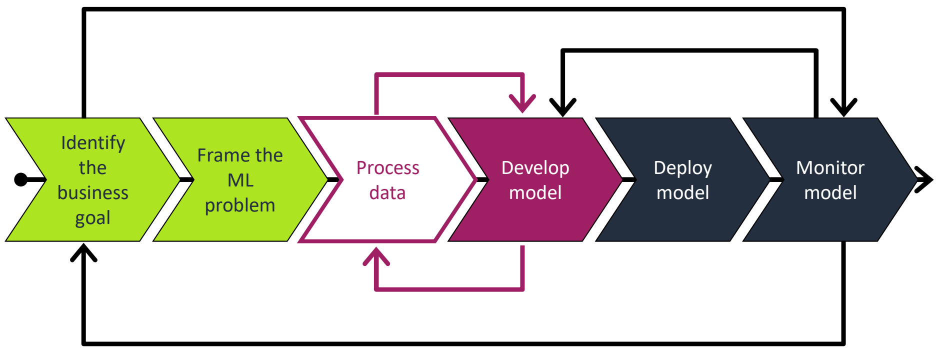
ML data concepts

months_as_customer	age	umbrella_limit	vehicle_claim	Is this fraud?
328	48	0	52080	
228	42	5000000	3510	
134	29	5000000	23100	
256	41	6000000	50720	
228	44	6000000	4550	

|Slide number 97

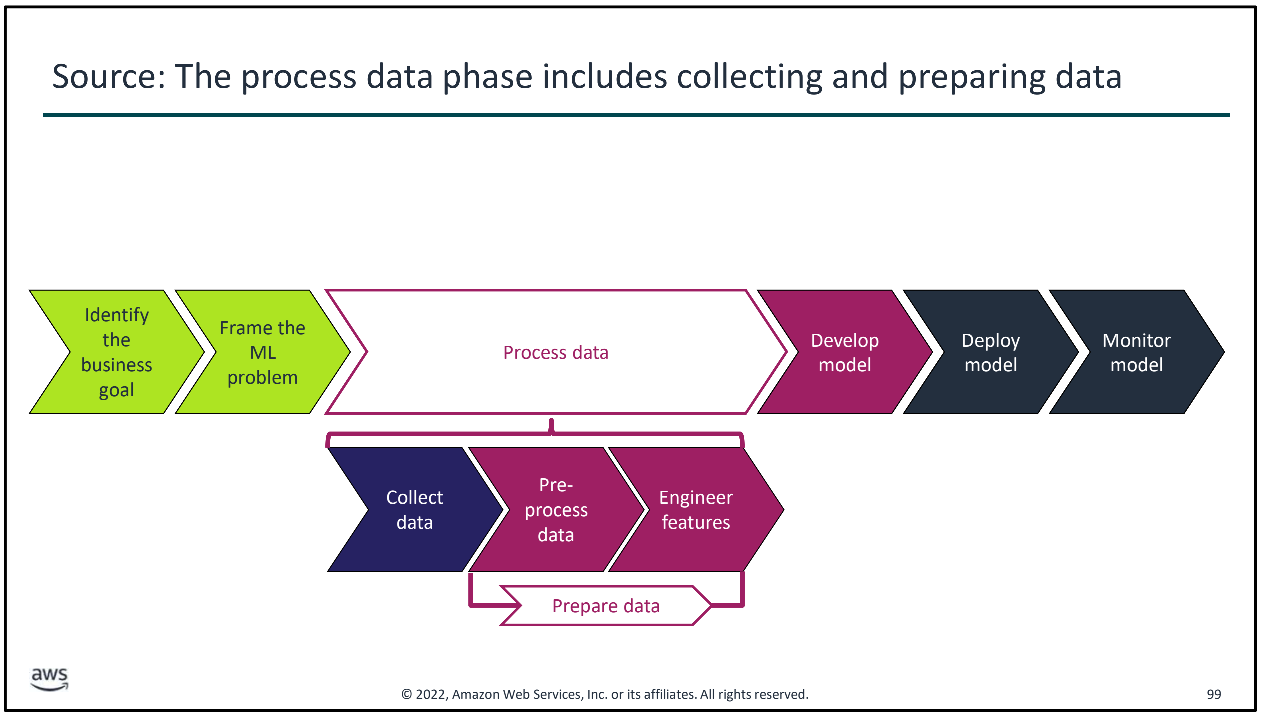


Source:The ML lifecycle defines a set of iterative phases

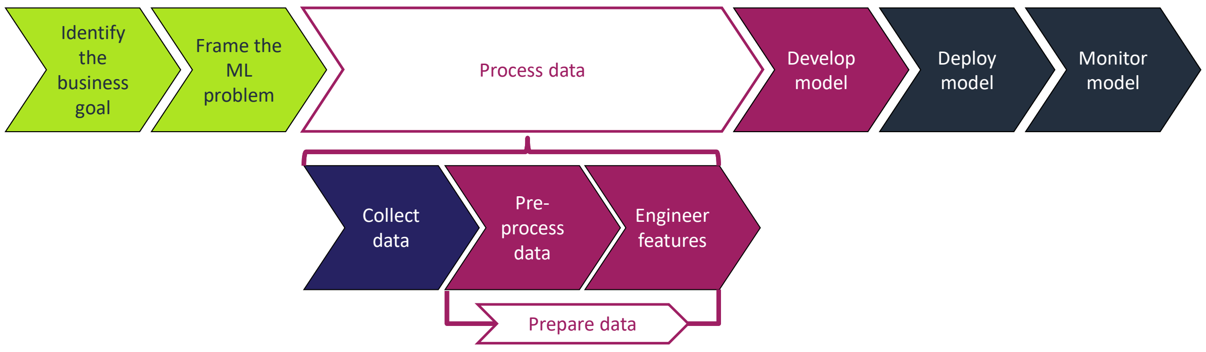


98

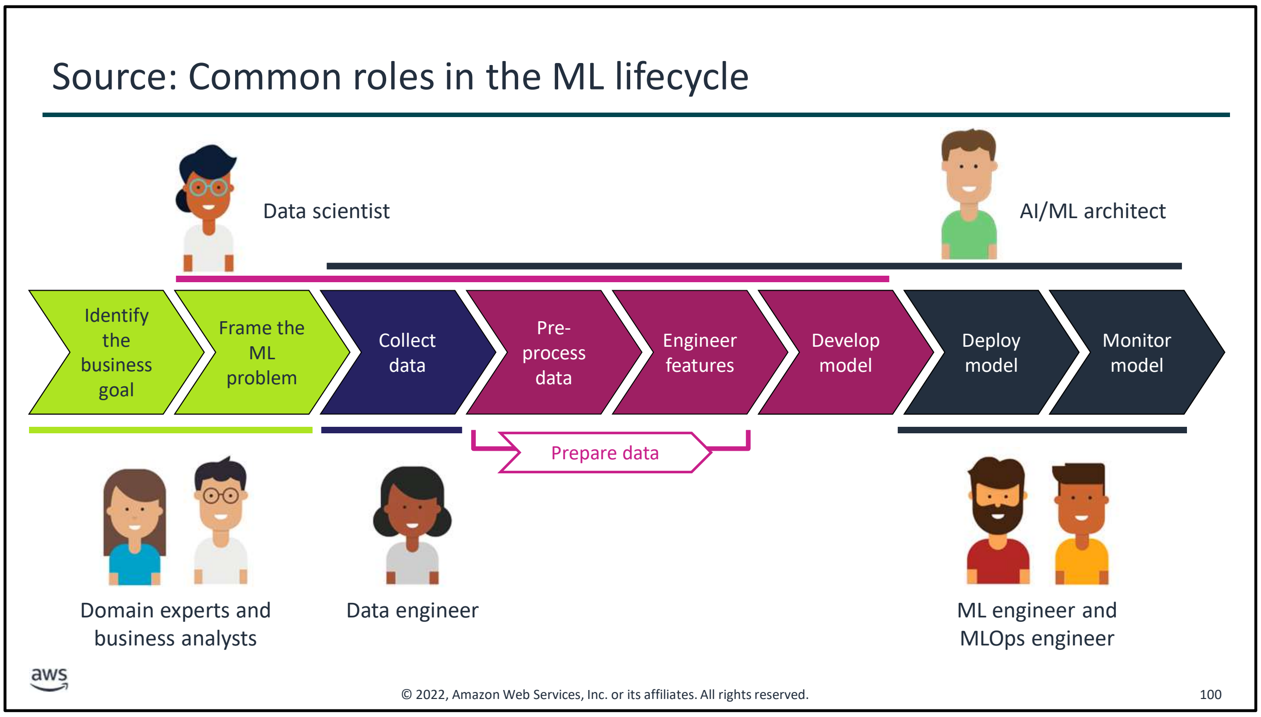
|Slide number 98



Source: The process data phase includes collecting and preparing data

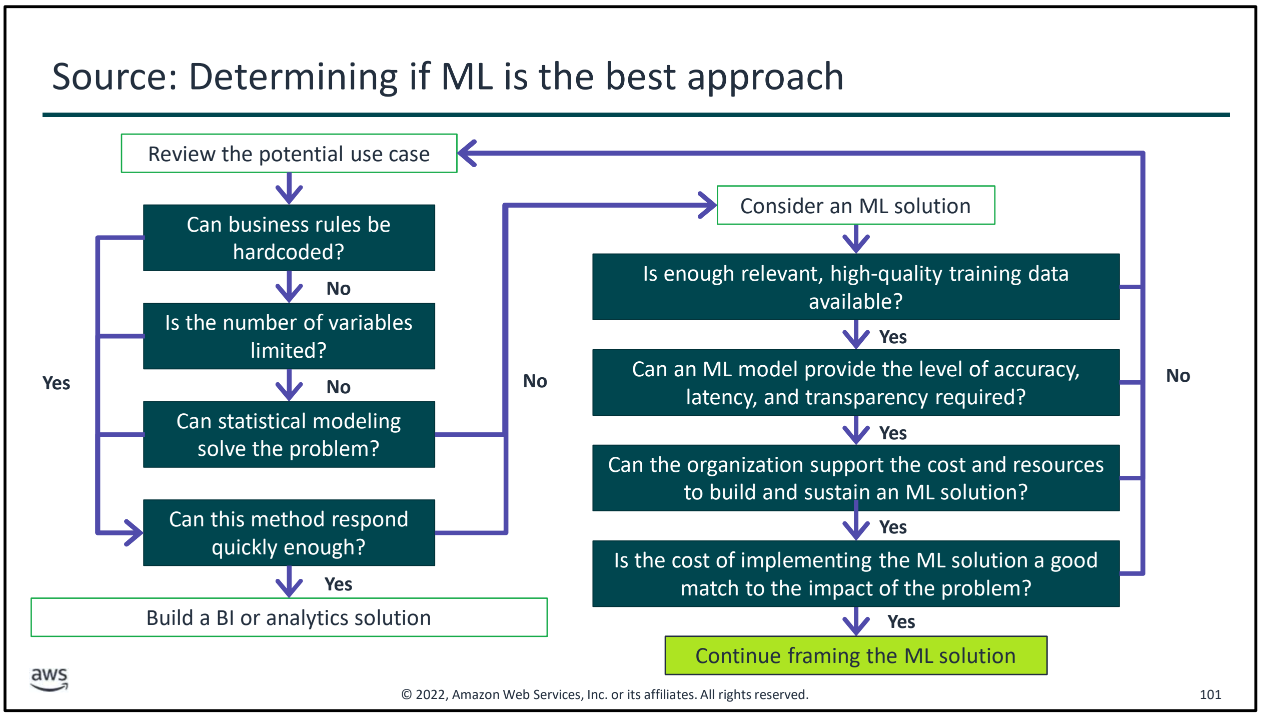


|Slide number 99



Source: Common roles in the ML lifecycle

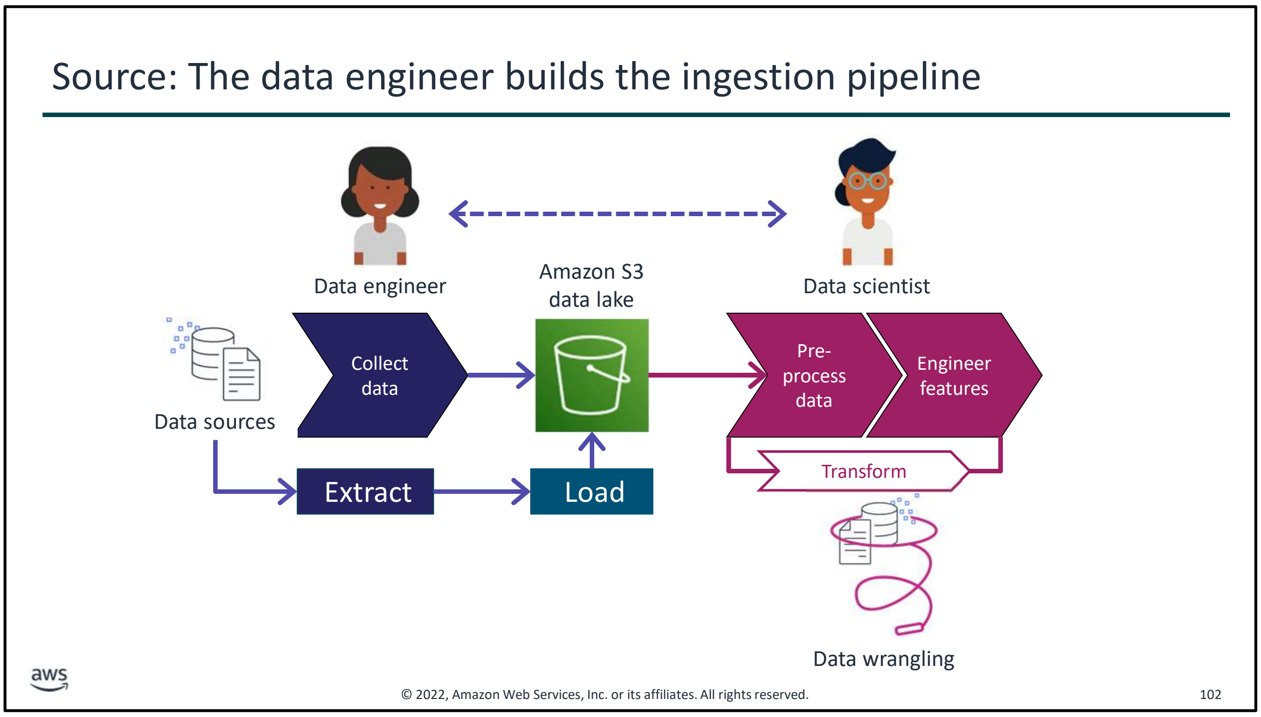
|Slide number 100



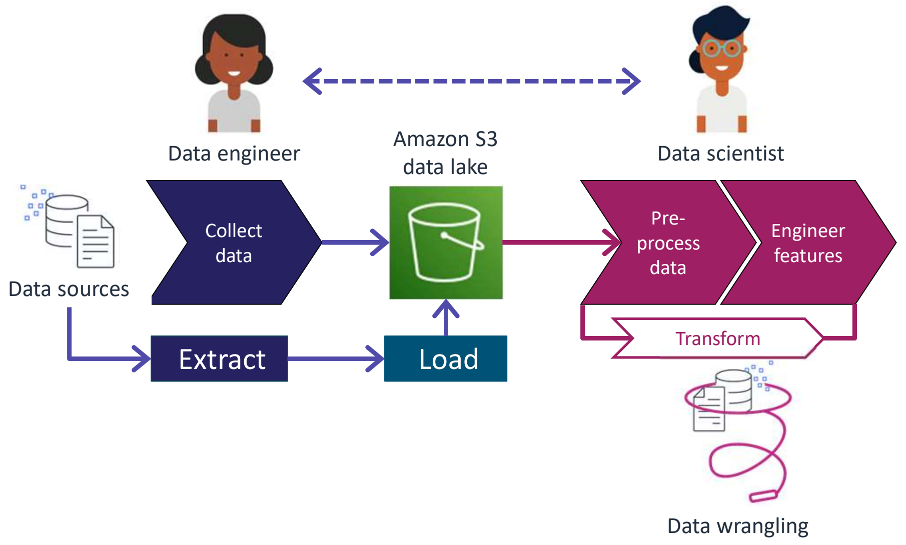
Source: Determining if ML is the best approach

Review the potential use case
Can business rules be hardcoded?
No
Is the number of variables limited?
No
Can statistical modeling solve the problem?
Can this method respond quickly enough?
Yes
Yes
Build a BI or analytics solution
No
Consider an ML solution
Is enough relevant, high-quality training data available?
Yes
Can an ML model provide the level of accuracy, latency, and transparency required?
Yes
Can the organization support the cost and resources to build and sustain an ML solution?
Yes
Is the cost of implementing the ML solution a good match to the impact of the problem?
Yes
Continue framing the ML solution
No

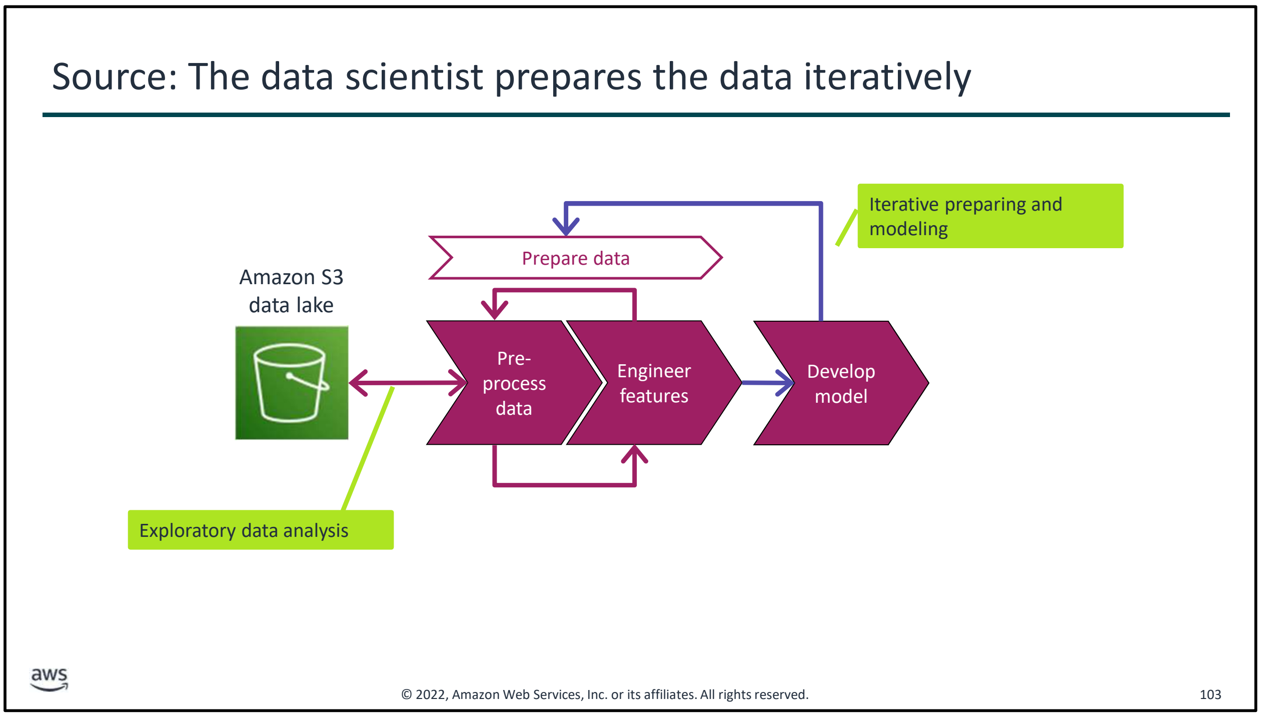
|Slide number 101



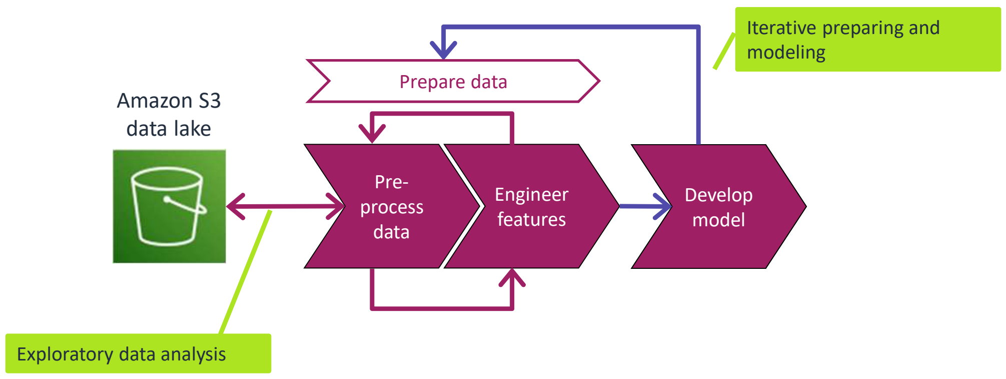
Source: The data engineer builds the ingestion pipeline



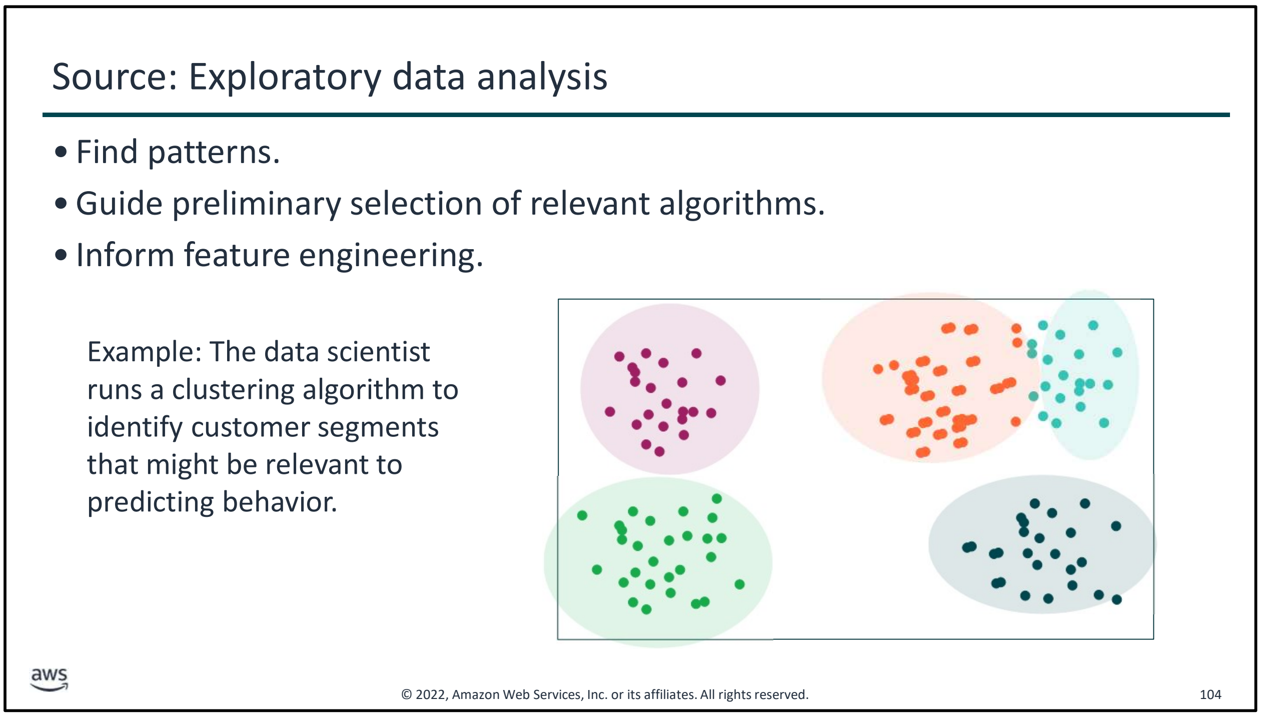
|Slide number 102



Source: The data scientist prepares the data iteratively



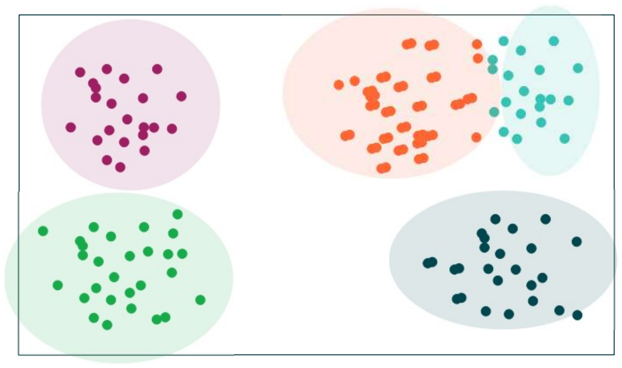
|Slide number 103



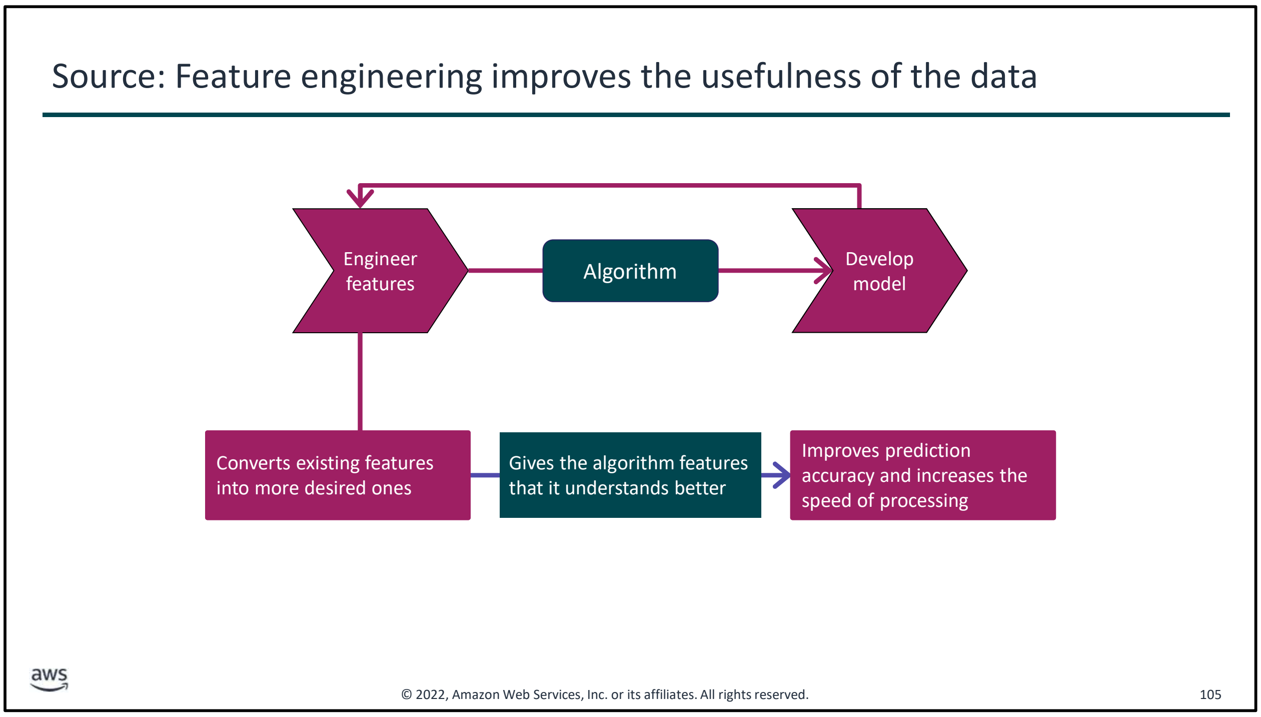
Source: Exploratory data analysis

- Find patterns.
- Guide preliminary selection of relevant algorithms.
- Inform feature engineering.

Example: The data scientist runs a clustering algorithm to identify customer segments that might be relevant to predicting behavior.



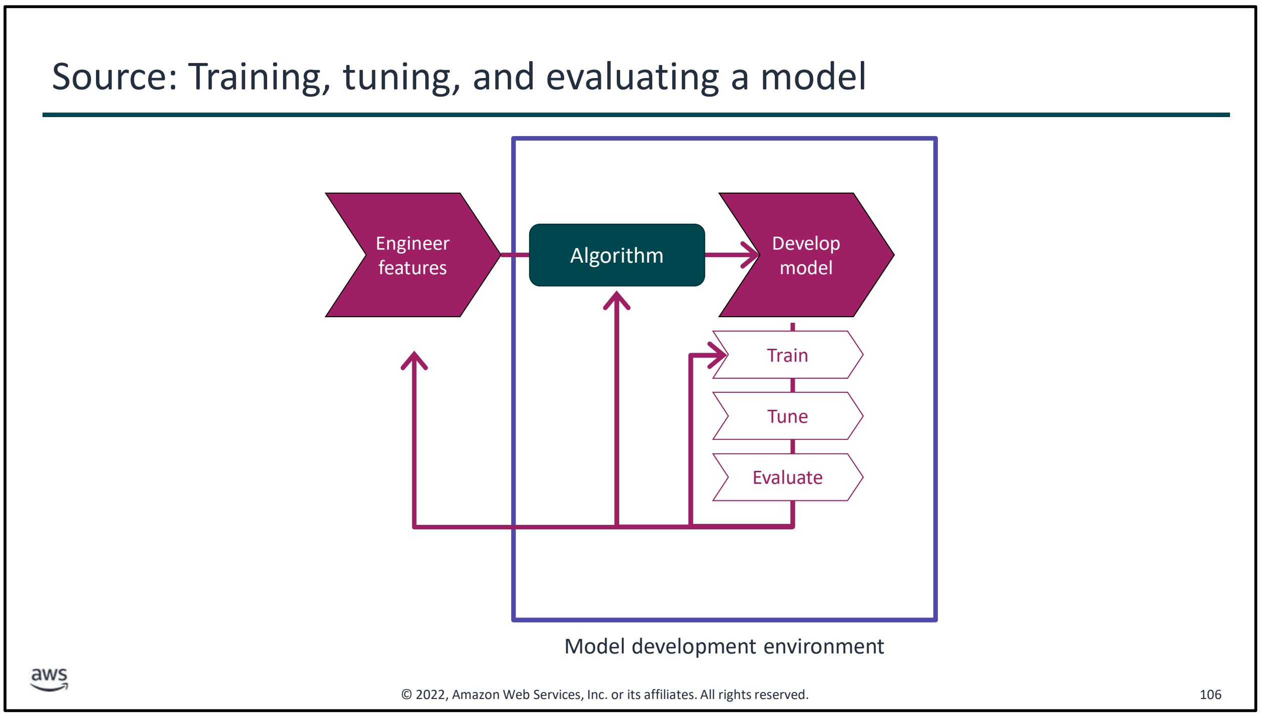
|Slide number 104



Source: Feature engineering improves the usefulness of the data

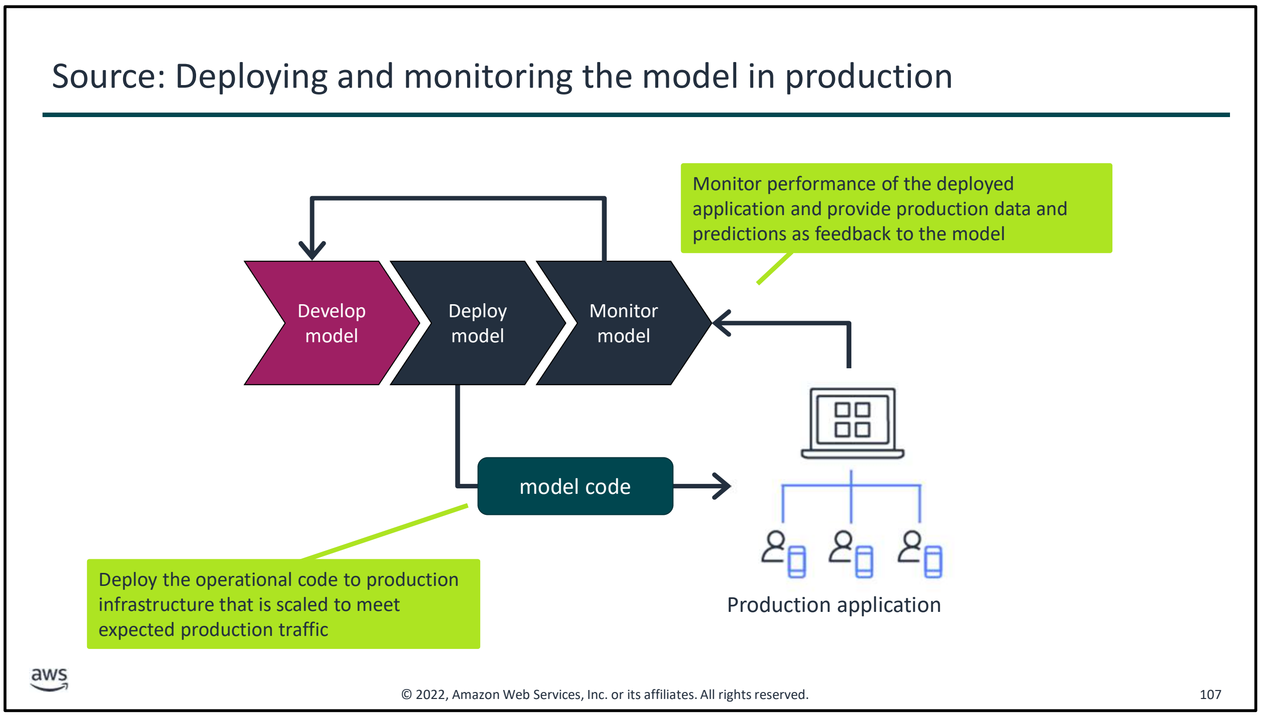
Engineer features
Algorithm
Develop model
Converts existing features into more desired ones
Gives the algorithm features that it understands better
Improves prediction accuracy and increases the speed of processing

|Slide number 105



Source: Training, tuning, and evaluating a model

|Slide number 106

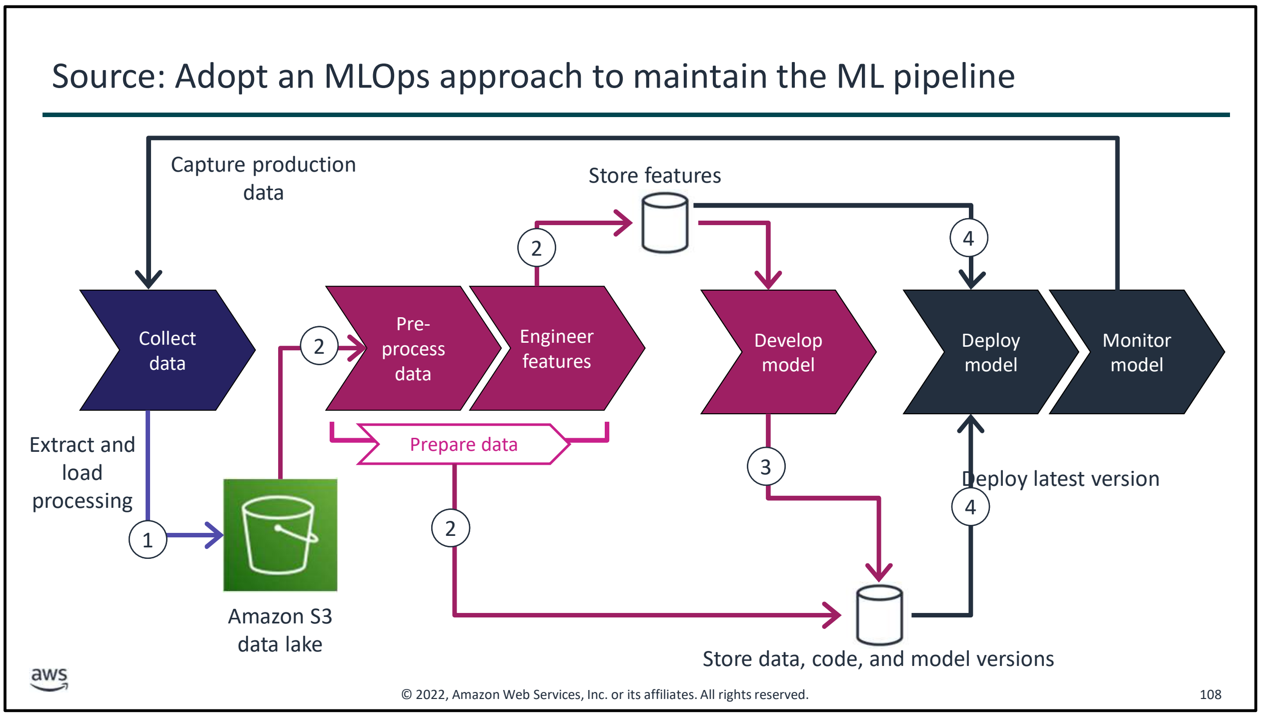


Source: Deploying and monitoring the model in production

Monitor performance of the deployed application and provide production data and predictions as feedback to the model

Deploy the operational code to production infrastructure that is scaled to meet expected production traffic

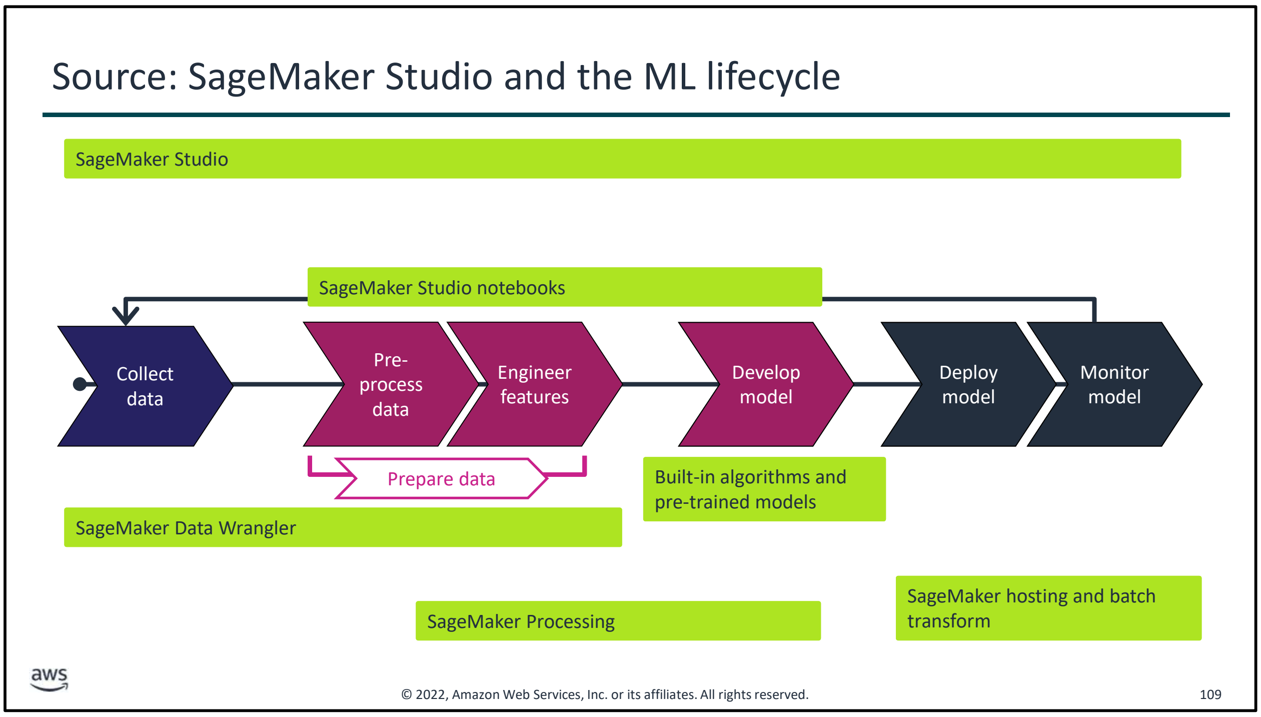
|Slide number 107



Source: Adopt an MLOps approach to maintain the ML pipeline

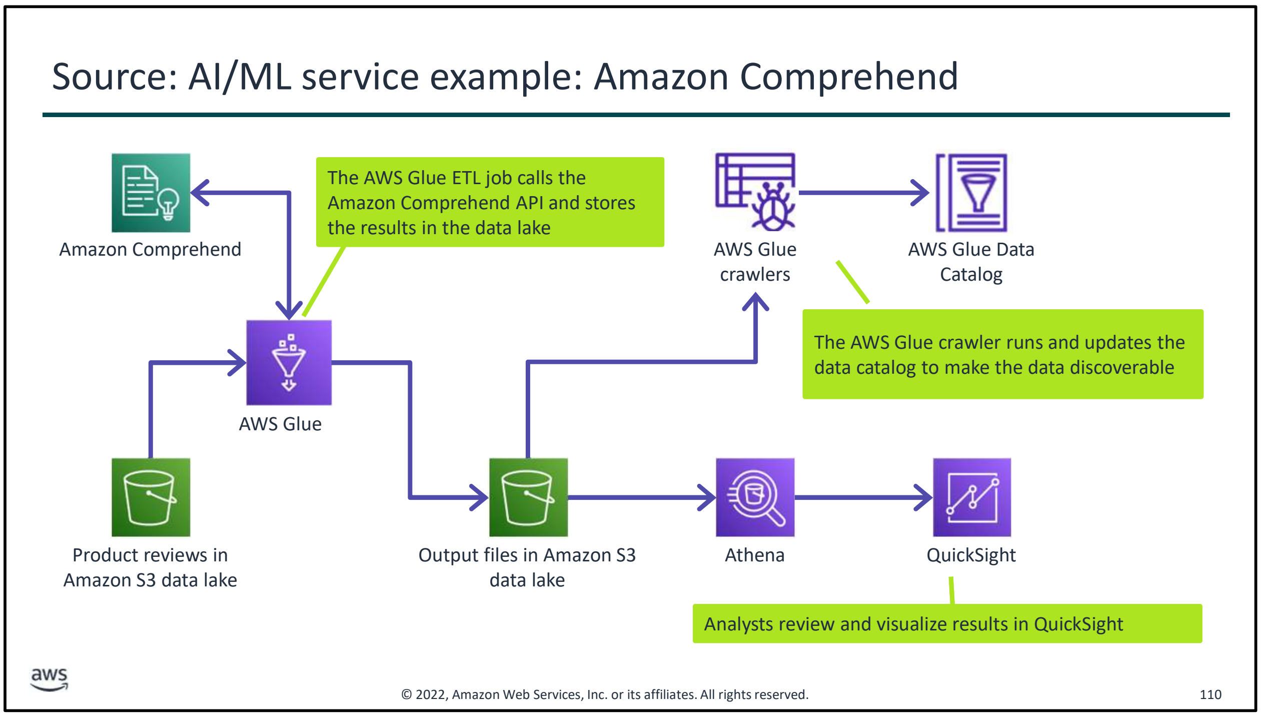
Capture production data
Store features
Collect data
Pre-process data
Engineer features
Develop model
Deploy model
Monitor model
Prepare data
Extract and load processing
Amazon S3 data lake
Deploy latest version
Store data, code, and model versions
1
2
2
2
3
4
4

|Slide number 108



Source: SageMaker Studio and the ML lifecycle

|Slide number 109

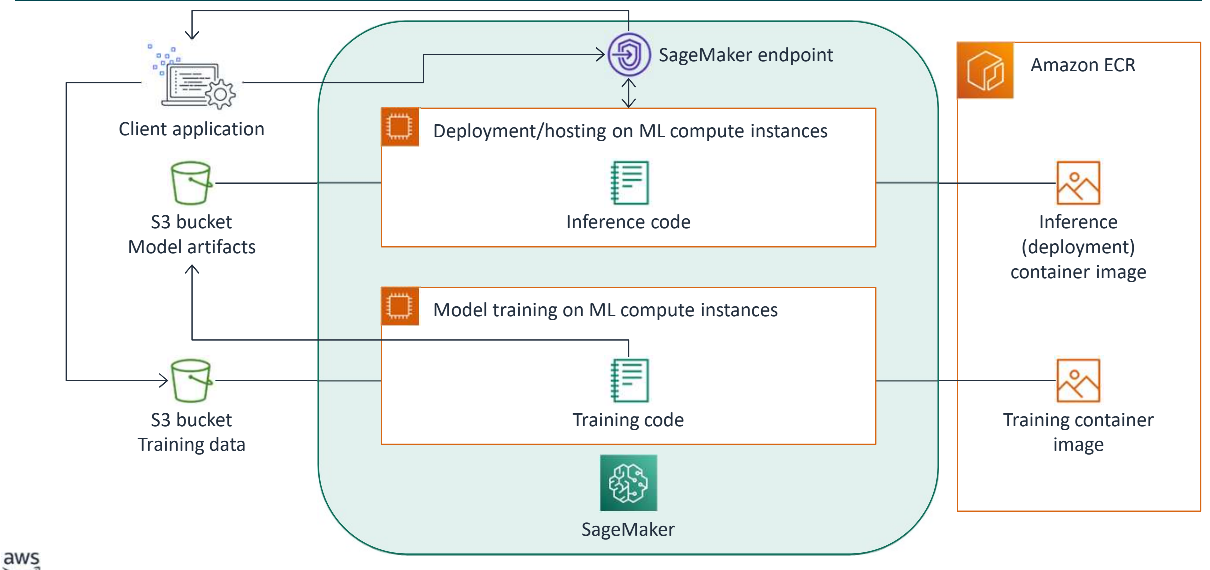


Source: AI/ML service example: Amazon Comprehend

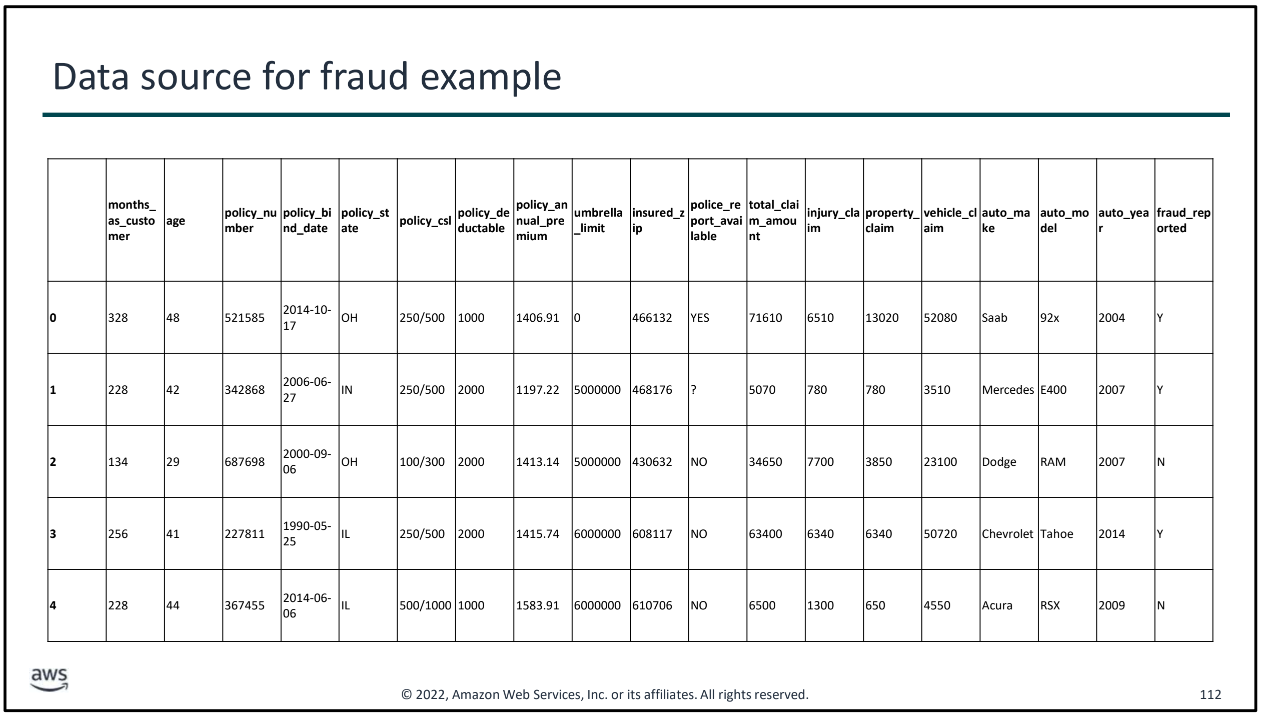
Amazon Comprehend
The AWS Glue ETL job calls the Amazon Comprehend API and stores the results in the data lake
AWS Glue crawlers
AWS Glue Data Catalog
The AWS Glue crawler runs and updates the data catalog to make the data discoverable
AWS Glue
Product reviews in Amazon S3 data lake
Output files in Amazon S3 data lake
Athena
QuickSight
Analysts review and visualize results in QuickSight

|Slide number 110

Source: Training and deploying a model with SageMaker



|Slide number 111



Data source for fraud example

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	police_report_available	total_claim_amount	injury_claim	property_claim	vehicle_claim	auto_make	auto_model	auto_year	fraud_reported
0	328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132	YES	71610	6510	13020	52080	Saab	92x	2004	Y
1	228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176	?	5070	780	780	3510	Mercedes	E400	2007	Y
2	134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632	NO	34650	7700	3850	23100	Dodge	RAM	2007	N
3	256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117	NO	63400	6340	6340	50720	Chevrolet	Tahoe	2014	Y
4	228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706	NO	6500	1300	650	4550	Acura	RSX	2009	N

|Slide number 112

Reference for fraud example