

Welcome to the introduction to machine learning lesson!

Machine learning is creating rapid and exciting changes across all levels of society.

- *It is the engine behind the recent advancements in industries such as autonomous vehicles.
- *It allows for more accurate and rapid translation of the text into hundreds of languages.
- *It powers the AI assistants you might find in your home.
- *It can help improve worker safety.
- *It can speed up drug design.

Machine learning is a complex subject area. Our goal in this lesson is to introduce you to some of the most common terms and ideas used in machine learning. I will then walk you through the different steps involved in machine learning and finish with a series of examples that use machine learning to solve real-world situations.

Let's look at how the lesson outline.

Lesson Outline

This lesson is divided into the following sections:

First, we'll discuss what machine learning is, common terminology, and common components involved in creating a machine learning project. Next, we'll step into the shoes of a machine learning practitioner.

Machine learning involves using trained models to generate predictions and detect patterns from data. To understand the process, we'll break down the different steps involved and examine a common process that applies to the majority of machine learning projects.

Finally, we'll take you through three examples using the steps we described to solve real-life scenarios that might be faced by machine learning practitioners.

What is Machine Learning?

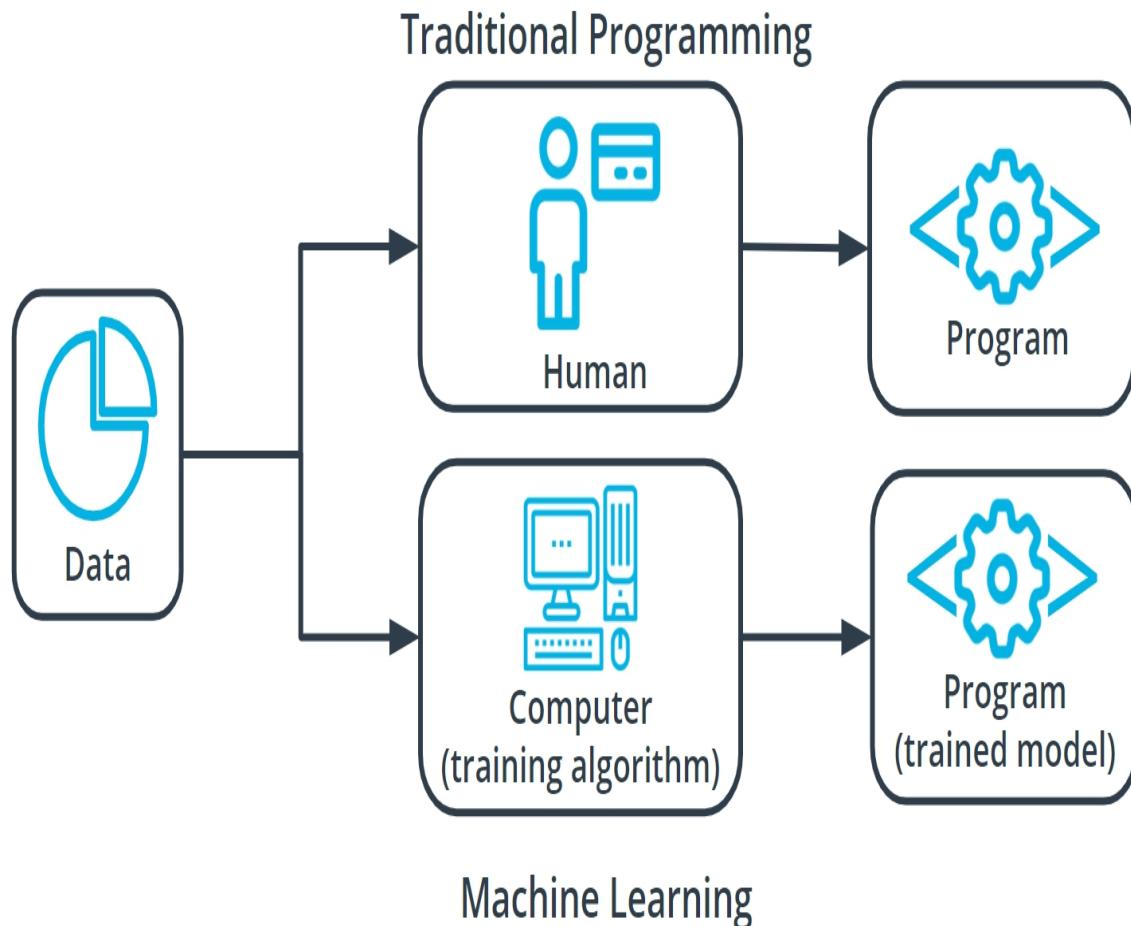
Machine learning (ML) is a modern software development technique and a type of artificial intelligence (AI) that enables computers to solve problems by using examples of real-world data. It allows computers to automatically learn and improve from experience without being explicitly programmed to do so.

Summary

Machine learning is part of the broader field of artificial intelligence. This field is concerned with the capability of machines to perform activities using human-like intelligence. Within machine learning there are several different kinds of tasks or techniques:

- In **supervised learning**, every training sample from the dataset has a corresponding label or output value associated with it. As a result, the algorithm learns to predict labels or output values. We will explore this in-depth in this lesson.
- In **unsupervised learning**, there are no labels for the training data. A machine learning algorithm tries to learn the underlying patterns or distributions that govern the data. We will explore this in-depth in this lesson.
- In **reinforcement learning**, the algorithm figures out which actions to take in a situation to maximize a reward (in the form of a number) on the way to reaching a specific goal. This is a completely different approach than supervised and unsupervised learning. We will dive deep into this in the next lesson.

How does machine learning differ from traditional programming-based approaches?



In traditional problem-solving with software, a person analyzes a problem and engineers a solution in code to solve that problem. For many real-world problems, this process can be laborious (or even impossible) because a correct solution would need to consider a vast number of edge cases.

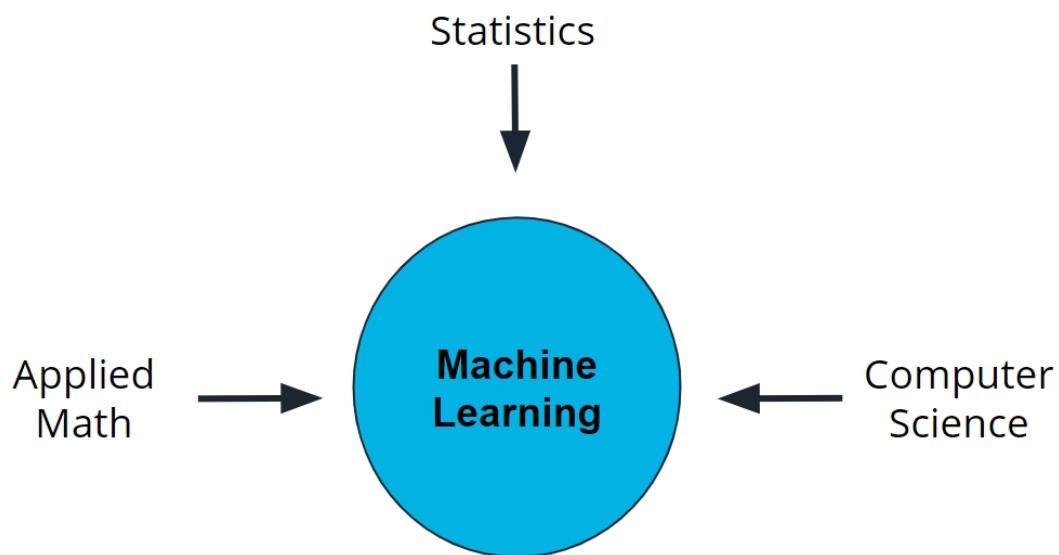
Imagine, for example, the challenging task of writing a program that can detect if a cat is present in an image. Solving this in the traditional way would require careful attention to details like varying lighting conditions, different types of cats, and various poses a cat might be in.

In machine learning, the problem solver abstracts away part of their solution as a flexible component called a *model*, and uses a special program called a *model training algorithm* to adjust that model to real-world data. The result is a trained model which can be used to predict outcomes that are not part of the data set used to train it.

In a way, machine learning automates some of the statistical reasoning and pattern-matching the problem solver would traditionally do.

The overall goal is to use a *model* created by a *model training algorithm* to generate predictions or find patterns in data that can be used to solve a problem.

Understanding Terminology



Machine learning is a new field created at the intersection of statistics, applied math, and computer science. Because of the rapid and recent growth of machine learning, each of these fields might use slightly different formal definitions of the same terms.

Terminology

Machine learning, or *ML*, is a modern software development technique that enables computers to solve problems by using examples of real-world data.

In **supervised learning**, every training sample from the dataset has a corresponding label or output value associated with it. As a result, the algorithm learns to predict labels or output values.

In **reinforcement learning**, the algorithm figures out which actions to take in a situation to maximize a reward (in the form of a number) on the way to reaching a specific goal.

In **unsupervised learning**, there are no labels for the training data. A machine learning algorithm tries to learn the underlying patterns or distributions that govern the data.

Components of Machine Learning



Machine Learning
Model

Model Training
Algorithm

Model Inference
Algorithm

Nearly all tasks solved with machine learning involve three primary components:

- A machine learning model
- A model training algorithm
- A model inference algorithm

Clay Analogy for Machine Learning

You can understand the relationships between these components by imagining the stages of crafting a teapot from a lump of clay.

1. First, you start with a block of raw clay. At this stage, the clay can be molded into many different forms and be used to serve many different purposes. You decide to use this lump of clay to make a teapot.
2. So how do you create this teapot? You inspect and analyze the raw clay and decide how to change it to make it look more like the teapot you have in mind.
3. Next, you mold the clay to make it look more like the teapot that is your goal. Congratulations! You've completed your teapot. You've inspected the materials, evaluated how to change them to reach your goal, and made the changes, and the teapot is now ready for your enjoyment.

What are machine learning models?

A machine learning model, like a piece of clay, can be molded into many different forms and serve many different purposes. A more technical definition would be that a machine learning model is a block of code or framework that can be modified to solve different but related problems based on the data provided.

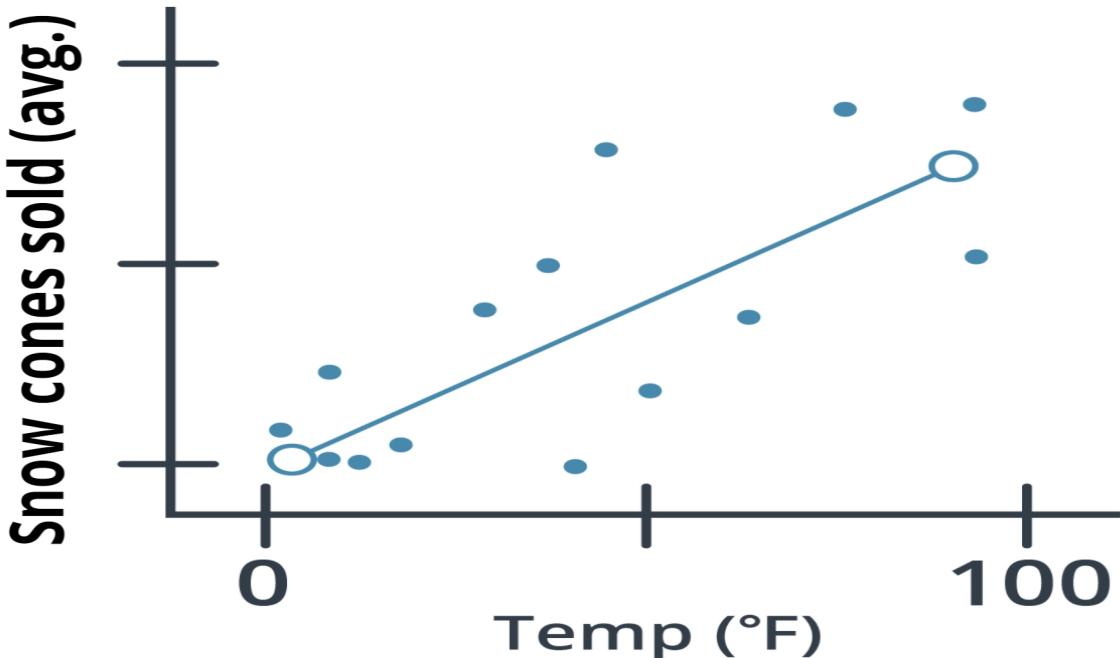
Important

A model is an extremely generic program(or block of code), made specific by the data used to train it. It is used to solve different problems.

Two simple examples

Example 1

Imagine you own a snow cone cart, and you have some data about the average number of snow cones sold per day based on the high temperature. You want to better understand this relationship to make sure you have enough inventory on hand for those high sales days.

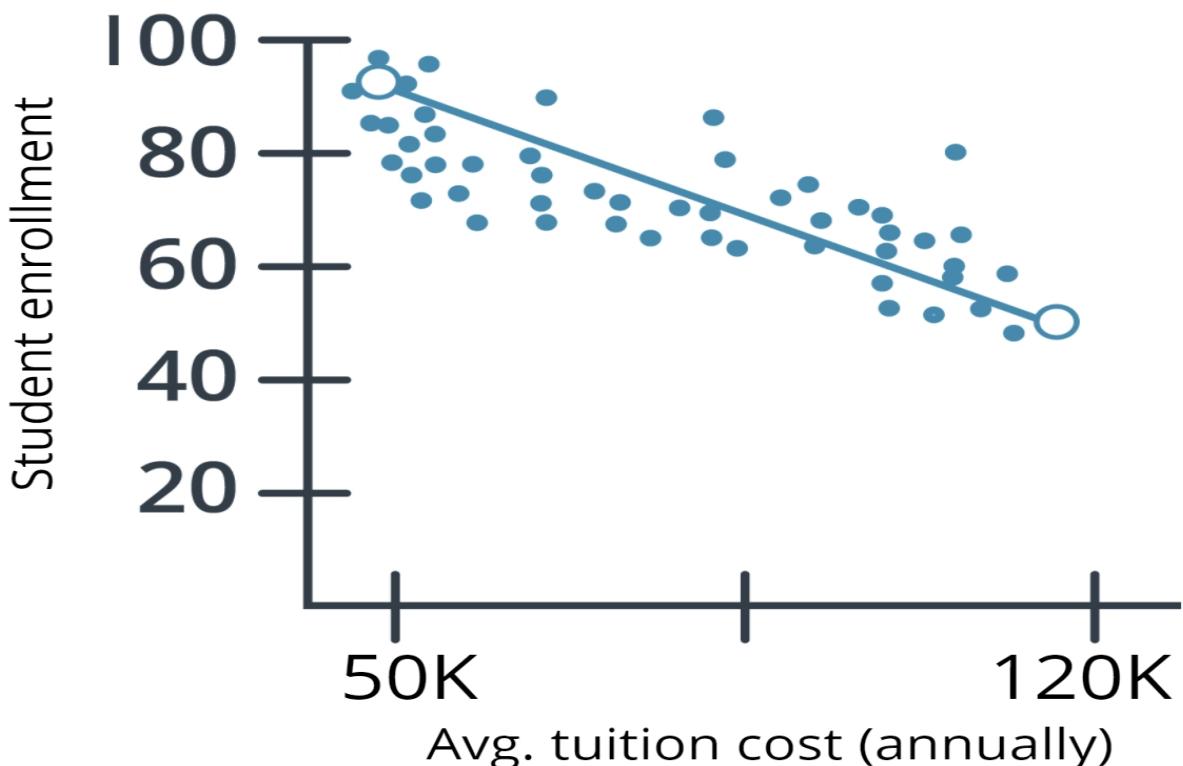


In the graph above, you can see one example of a model, a *linear regression model* (indicated by the solid line). You can see that, based on the data provided, the model predicts that as the high temperate for the day *increases* so do the average number of snow cones sold. Sweet!

Example 2

Let's look at a different example that uses the same *linear regression model*, but with different data and to answer completely different questions.

Imagine that you work in higher education and you want to better understand the relationship between the cost of enrollment and the number of students attending college. In this example, our model predicts that as the cost of tuition increases the number of people attending college is likely to decrease.



Using the same linear regression model (indicated by the solid line), you can see that the number of people attending college does go down as the cost increases.

Both examples showcase that a model is a generic program made specific by the data used to train it.

Model Training

How are model training algorithms used to train a model?

In the preceding section, we talked about two key pieces of information: a model and data. In this section, we show you how those two pieces of information are used to create a trained model. This process is called *model training*.

Model training algorithms work through an interactive process

Let's revisit our clay teapot analogy. We've gotten our piece of clay, and now we want to make a teapot. Let's look at the algorithm for molding clay and how it resembles a machine learning algorithm:

- **Think about the changes that need to be made.** The first thing you would do is inspect the raw clay and think about what changes can be made to make it look more like a teapot. Similarly, a model training algorithm uses the model to process data and then compares the results against some end goal, such as our clay teapot.
- **Make those changes.** Now, you mold the clay to make it look more like a teapot. Similarly, a model training algorithm gently nudges specific parts of the model in a direction that brings the model closer to achieving the goal.
- **Repeat.** By iterating over these steps over and over, you get closer and closer to what you want until you determine that you're close enough that you can stop.



Model Inference: Using Your Trained Model

Now you have our completed teapot. You inspected the clay, evaluated the changes that needed to be made, and made them, and now the teapot is ready for you to use. Enjoy your tea!

So what does this mean from a machine learning perspective? We are ready to use the model inference algorithm to generate predictions using the trained model. This process is often referred to as **model inference**.



Images

- [Clay](#)
- [Clay hands](#)
- [Teapot](#)

Terminology

A **model** is an extremely generic program, made specific by the data used to train it.

Model training algorithms work through an interactive process where the current model iteration is analyzed to determine what changes can be made to get closer to the goal. Those changes are made and the iteration continues until the model is evaluated to meet the goals.

Model inference is when the trained model is used to generate predictions.

Introduction to the Five Machine Learning Steps



Major Steps in the Machine Learning Process

In the preceding diagram, you can see an outline of the major steps of the machine learning process. Regardless of the specific model or training algorithm used, machine learning practitioners practice a common workflow to accomplish machine learning tasks.

These steps are iterative. In practice, that means that at each step along the process, you review how the process is going. Are things operating as you expected? If not, go back and revisit your current step or previous steps to try and identify the breakdown.

Step One: Define the Problem

How do You Start a Machine Learning Task?

- **Define a very specific task.**
Think back to the snow cone sales example. Now imagine that you own a frozen treats store and you sell snow cones along with many other products. You wonder, "How do I increase sales?" It's a valid question, but it's the **opposite** of a very specific task. The following examples demonstrate how a machine learning practitioner might attempt to answer that question.
 - "Does adding a \$1.00 charge for sprinkles on a hot fudge sundae increase the sales of hot fudge sundaes?"
 - "Does adding a \$0.50 charge for organic flavors in your snow cone increase the sales of snow cones?"
- **Identify the machine learning task we might use to solve this problem.**
This helps you better understand the data you need for a project.

What is a Machine Learning Task?

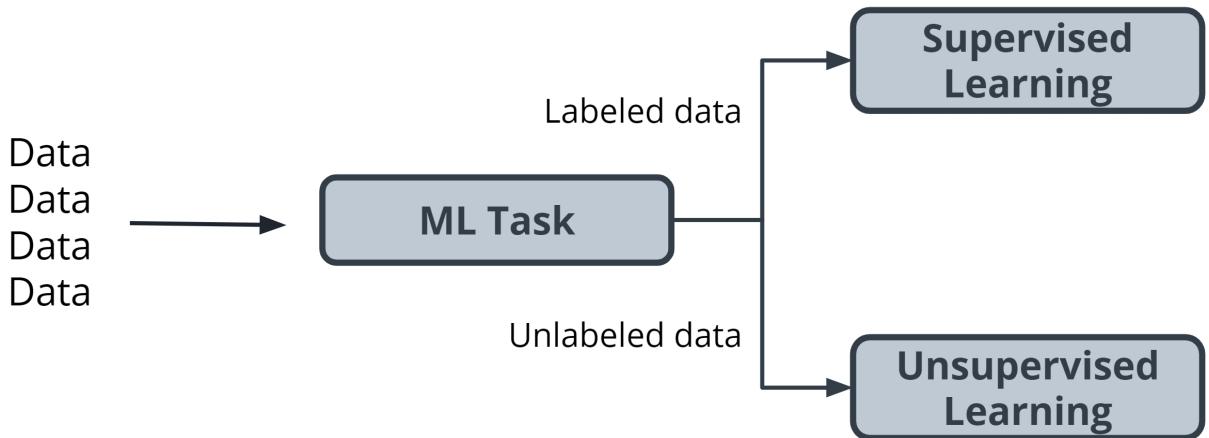
All model training algorithms, and the models themselves, take data as their input. Their outputs can be very different and are classified into a few different groups based on the *task* they are designed to solve. Often, we use the kind of data required to train a model as part of defining a machine learning task.

In this lesson, we will focus on two common machine learning tasks:

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

Supervised and Unsupervised Learning

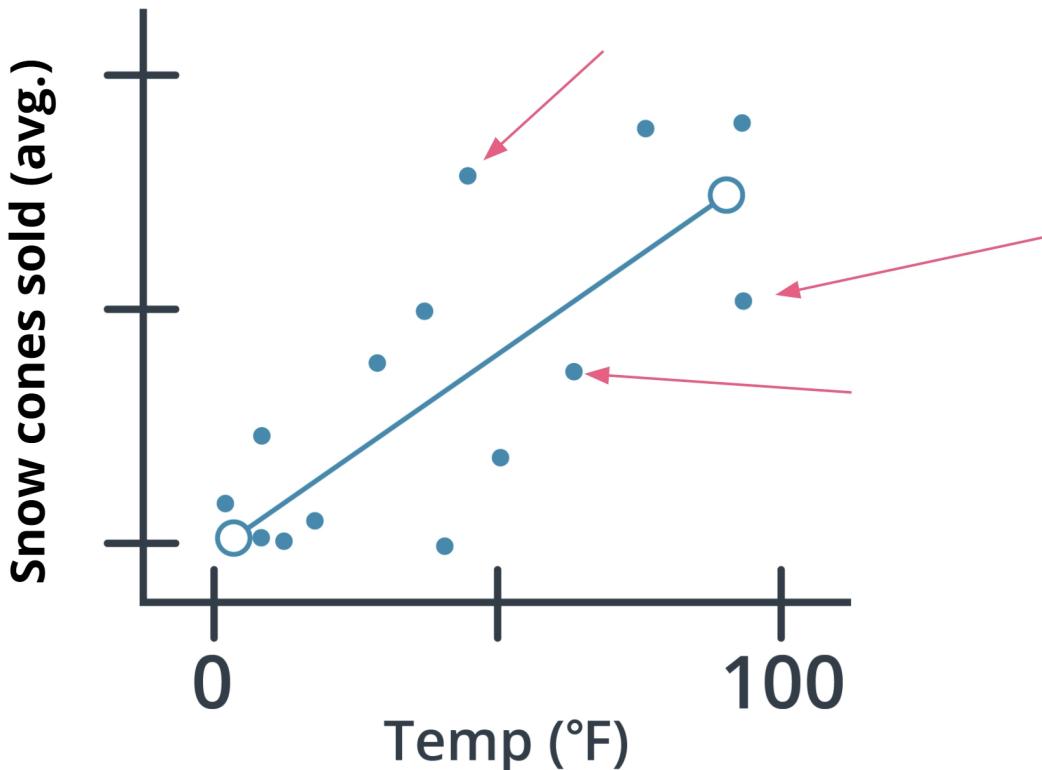
The presence or absence of labeling in your data is often used to identify a machine learning task.



Supervised tasks

A task is *supervised* if you are using labeled data. We use the term *labeled* to refer to data that already contains the solutions, called *labels*.

For example: Predicting the number of snow cones sold based on the temperatures is an example of supervised learning.



In the preceding graph, the data contains both a temperature and the number of snow cones sold. Both components are used to generate the linear regression shown on the graph. Our goal was to predict the number of snow cones sold, and we feed that value into the model. We are providing the model with labeled data and therefore, we are performing a *supervised machine learning task*.

Unsupervised tasks

A task is considered to be *unsupervised* if you are using *unlabeled data*. This means you don't need to provide the model with any kind of label or solution while the model is being trained.

Let's take a look at unlabeled data.



Image credit: [Unsplash](#)

- Take a look at the preceding picture. Did you notice the tree in the picture? What you just did, when you noticed the object in the picture and identified it as a tree, is called *labeling the picture*. Unlike you, a computer just sees that image as a matrix of pixels of varying intensity.
- Since this image does not have the labeling in its original data, it is considered *unlabeled*.

How do we classify tasks when we don't have a label?

Unsupervised learning involves using data that doesn't have a label. One common task is called **clustering**. Clustering helps to determine if there are any naturally occurring groupings in the data.

Let's look at an example of how clustering in unlabeled data works.

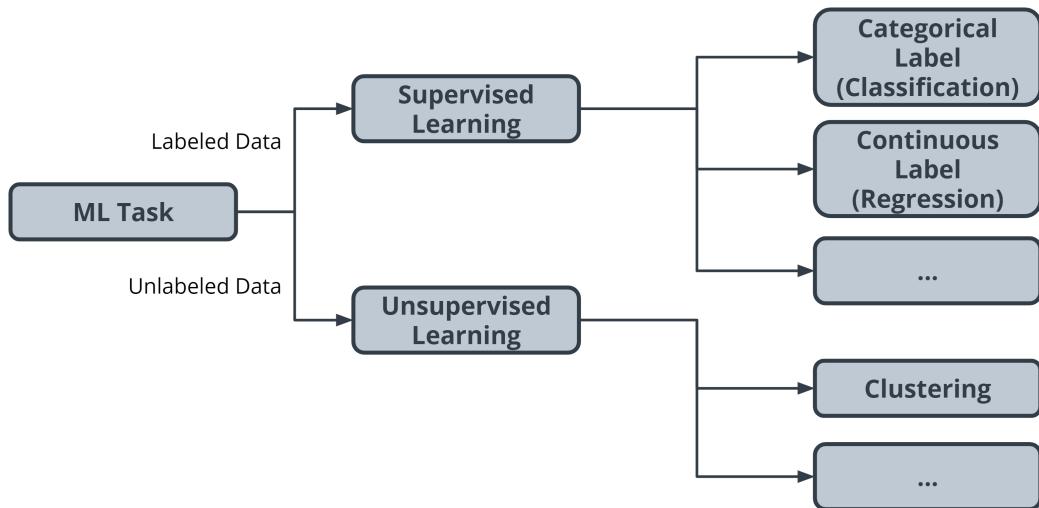
Identifying book micro-genres with unsupervised learning

Imagine that you work for a company that recommends books to readers.

The assumption: You are fairly confident that micro-genres exist, and that there is one called *Teen Vampire Romance*. Because you don't know which micro-genres exist, you can't use **supervised learning** techniques.

This is where the **unsupervised learning** clustering technique might be able to detect some groupings in the data. The words and phrases used in the book description might provide some guidance on a book's micro-genre.

Further Classifying by using Label Types



Initially, we divided tasks based on the presence or absence of labeled data while training our model. Often, tasks are further defined by the type of label which is present.

In **supervised** learning, there are two main identifiers you will see in machine learning:

- A **categorical** label has a discrete set of possible values. In a machine learning problem in which you want to identify the type of flower based on a picture, you would train your model using images that have been labeled with the categories of flower you would want to identify. Furthermore, when you work with categorical labels, you often carry out classification tasks*, which are part of the supervised learning family.
 - A **continuous** (regression) label does not have a discrete set of possible values, which often means you are working with numerical data. In the snow cone sales example, we are trying to predict the number* of snow cones sold. Here, our label is a number that could, in theory, be any value.
- In unsupervised learning, **clustering** is just one example. There are many other options, such as deep learning.

Terminology

- **Clustering.** Unsupervised learning task that helps to determine if there are any naturally occurring groupings in the data.
- A **categorical label** has a discrete set of possible values, such as "is a cat" and "is not a cat."
- A **continuous (regression) label** does not have a discrete set of possible values, which means possibly an unlimited number of possibilities.
- **Discrete:** A term taken from statistics referring to an outcome taking on only a finite number of values (such as days of the week).
- A **label** refers to data that already contains the solution.
- Using **unlabeled** data means you don't need to provide the model with any kind of label or solution while the model is being trained.

Step Two: Build a Dataset

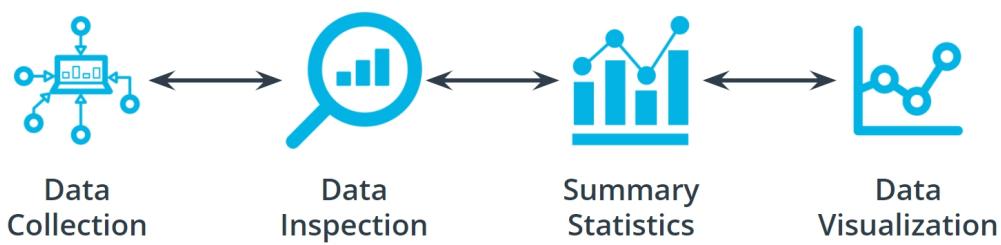
Summary

The next step in the machine learning process is to build a dataset that can be used to solve your machine learning-based problem. Understanding the data needed helps you select better models and algorithms so you can build more effective solutions.

The most important step of the machine learning process

Working with data is perhaps the most overlooked—yet most important—step of the machine learning process. In 2017, an O'Reilly study showed that machine learning practitioners spend 80% of their time working with their data.

The Four Aspects of Working with Data



You can take an entire class just on working with, understanding, and processing data for machine learning applications. Good, high-quality data is essential for any kind of machine learning project. Let's explore some of the common aspects of working with data.

Data collection

Data collection can be as straightforward as running the appropriate SQL queries or as complicated as building custom web scraper applications to collect data for your project. You might even have to run a model over your data to generate needed labels. Here is the fundamental question:

Does the data you've collected match the machine learning task and problem you have defined?

Data inspection

The quality of your data will ultimately be the largest factor that affects how well you can expect your model to perform. As you inspect your data, look for:

- Outliers
- Missing or incomplete values
- Data that needs to be transformed or preprocessed so it's in the correct format to be used by your model

Summary statistics

Models can assume how your data is structured.

Now that you have some data in hand it is a good best practice to check that your data is in line with the underlying assumptions of your chosen machine learning model.

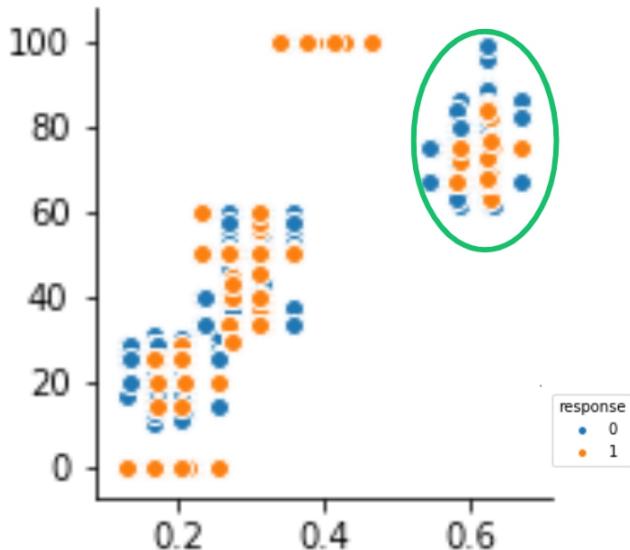
With many statistical tools, you can calculate things like the mean, inner-quartile range (IQR), and standard deviation. These tools can give you insight into the *scope*, *scale*, and *shape* of the dataset.

Data visualization

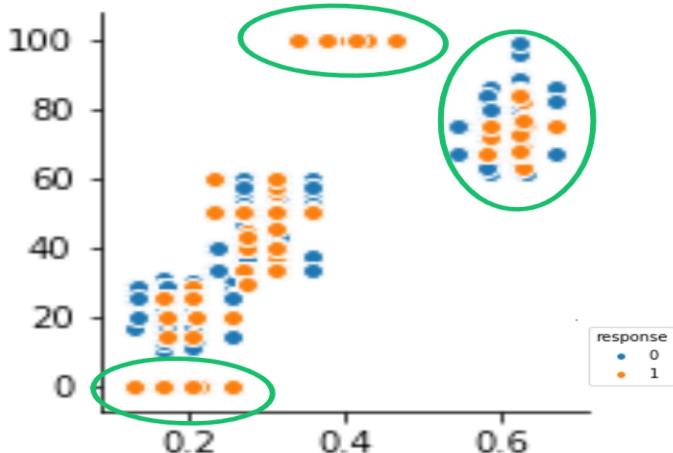
You can use data visualization to see outliers and trends in your data and to help stakeholders understand your data.

Look at the following two graphs. In the first graph, some data seems to have clustered into different groups. In the second graph, some data points might be outliers.

Some of the data seems to cluster in groups



Some of the data points seem to be outlier.



Terminology

- *Impute* is a common term referring to different statistical tools which can be used to calculate missing values from your dataset.
- *Outliers* are data points that are significantly different from others in the same sample.

Additional reading

- In machine learning, you use several statistical-based tools to better understand your data. The `sklearn` library has many examples and tutorials, such as this example demonstrating [outlier detection on a real dataset](#).

Step Three: Model Training

You're ready to start training your first model.

Splitting your Dataset

The first step in model training is to randomly split the dataset. This allows you to keep some data hidden during training, so that data can be used to evaluate your model before you put it into production. Specifically, you do this to test against the bias-variance trade-off. If you're interested in learning more, see the [Further learning and reading](#) section.

Splitting your dataset gives you two sets of data:

- *Training dataset*: The data on which the model will be trained. Most of your data will be here. Many developers estimate about 80%.
- *Test dataset*: The data withheld from the model during training, which is used to test how well your model will generalize to new data.

Model Training Terminology

The model training algorithm iteratively updates a model's parameters to minimize some loss function.

Let's define those two terms:

- **Model parameters:** Model parameters are settings or configurations the training algorithm can update to change how the model behaves. Depending on the context, you'll also hear other more specific terms used to describe model parameters such as *weights* and *biases*. Weights, which are values that change as the model learns, are more specific to neural networks.
- **Loss function:** A loss function is used to codify the model's distance from this goal. For example, if you were trying to predict a number of snow cone sales based on the day's weather, you would care about making predictions that are as accurate as possible. So you might define a loss function to be "the average distance between your model's predicted number of snow cone sales and the correct number." You can see in the snow cone example this is the difference between the two purple dots.

Putting it All Together

The end-to-end training process is

- Feed the training data into the model.
- Compute the loss function on the results.
- Update the model parameters in a direction that reduces loss.

You continue to cycle through these steps until you reach a predefined stop condition. This might be based on a training time, the number of training cycles, or an even more intelligent or application-aware mechanism.

Advice From the Experts

Remember the following advice when training your model.

1. Practitioners often use machine learning frameworks that already have working implementations of models and model training algorithms. You could implement these from scratch, but you probably won't need to do so unless you're developing new models or algorithms.
2. Practitioners use a process called *model selection* to determine which model or models to use. The list of established models is constantly growing, and even seasoned machine learning practitioners may try many different types of models while solving a problem with machine learning.
3. *Hyperparameters* are settings on the model which are not changed during training but can affect how quickly or how reliably the model trains, such as the number of clusters the model should identify.
4. Be prepared to iterate.

Pragmatic problem solving with machine learning is rarely an exact science, and you might have assumptions about your data or problem which turn out to be false. Don't get discouraged. Instead, foster a habit of trying new things, measuring success, and comparing results across iterations.

Extended Learning

This information hasn't been covered in the above video but is provided for the advanced reader.

Linear models

One of the most common models covered in introductory coursework, linear models simply describe the relationship between a set of input numbers and a set of output numbers through a linear function (think of $y = mx + b$ or a line on a x vs y chart).

Classification tasks often use a strongly related logistic model, which adds an additional transformation mapping the output of the linear function to the range $[0, 1]$, interpreted as "probability of being in the target class." Linear models are fast to train and give you a great baseline against which to compare more complex models. A lot of media buzz is given to more complex models, but for most new problems, consider starting with a simple model.

Tree-based models

Tree-based models are probably the second most common model type covered in introductory coursework. They learn to categorize or regress by building an extremely large structure of nested *if/else blocks*, splitting the world into different regions at each if/else block. Training determines exactly where these splits happen and what value is assigned at each leaf region.

For example, if you're trying to determine if a light sensor is in sunlight or shadow, you might train tree of depth 1 with the final learned configuration being something like *if (sensor_value > 0.698), then return 1;*

`else return 0;`. The tree-based model XGBoost is commonly used as an off-the-shelf implementation for this kind of model and includes enhancements beyond what is discussed here. Try tree-based models to quickly get a baseline before moving on to more complex models.

Deep learning models

Extremely popular and powerful, deep learning is a modern approach based around a conceptual model of how the human brain functions. The model (also called a *neural network*) is composed of collections of *neurons* (very simple computational units) connected together by *weights* (mathematical representations of how much information to allow to flow from one neuron to the next). The process of training involves finding values for each weight.

Various neural network structures have been determined for modeling different kinds of problems or processing different kinds of data.

A short (but not complete!) list of noteworthy examples includes:

- **FFNN:** The most straightforward way of structuring a neural network, the Feed Forward Neural Network (FFNN) structures neurons in a series of layers, with each neuron in a layer containing weights to all neurons in the previous layer.
- **CNN:** Convolutional Neural Networks (CNN) represent nested filters over grid-organized data. They are by far the most commonly used type of model when processing images.
- **RNN/LSTM:** Recurrent Neural Networks (RNN) and the related Long Short-Term Memory (LSTM) model types are structured to effectively represent *for loops* in traditional computing, collecting state while iterating over some object. They can be used for processing sequences of data.
- **Transformer:** A more modern replacement for RNN/LSTMs, the transformer architecture enables training over larger datasets involving sequences of data.

Machine Learning Using Python Libraries

- For more classical models (linear, tree-based) as well as a set of common ML-related tools, take a look at `scikit-learn`. The web documentation for this library is also organized for those getting familiar with space and can be a great place to get familiar with some extremely useful tools and techniques.
- For deep learning, `mxnet`, `tensorflow`, and `pytorch` are the three most common libraries. For the purposes of the majority of machine learning needs, each of these is feature-paired and equivalent.

Terminology

Hyperparameters are settings on the model which are not changed during training but can affect how quickly or how reliably the model trains, such as the number of clusters the model should identify.

A **loss function** is used to codify the model's distance from this goal

Training dataset: The data on which the model will be trained. Most of your data will be here.

Test dataset: The data withheld from the model during training, which is used to test how well your model will generalize to new data.

Model parameters are settings or configurations the training algorithm can update to change how the model behaves.

Additional reading

- The Wikipedia entry on the [bias-variance](#) trade-off can help you understand more about this common machine learning concept.
- In this [AWS Machine Learning blog post](#), you can see how to train a machine-learning algorithm to predict the impact of weather on air quality using Amazon SageMaker.

Step Four: Model Evaluation

After you have collected your data and trained a model, you can start to evaluate how well your model is performing. The metrics used for evaluation are likely to be very specific to the problem you have defined. *As you grow in your understanding of machine learning, you will be able to explore a wide variety of metrics that can enable you to evaluate effectively.*

Using Model Accuracy

Model accuracy is a fairly common evaluation metric. *Accuracy* is the fraction of predictions a model gets right.

Here's an example:



Imagine that you built a model to identify a flower as one of two common species based on measurable details like petal length. You want to know how often your model predicts the correct species. This would require you to look at your model's accuracy.

Extended Learning

This information hasn't been covered in the above video but is provided for the advanced reader.

Using Log Loss

Log loss seeks to calculate how *uncertain* your model is about the predictions it is generating. In this context, uncertainty refers to how likely a model thinks the predictions being generated are to be correct.

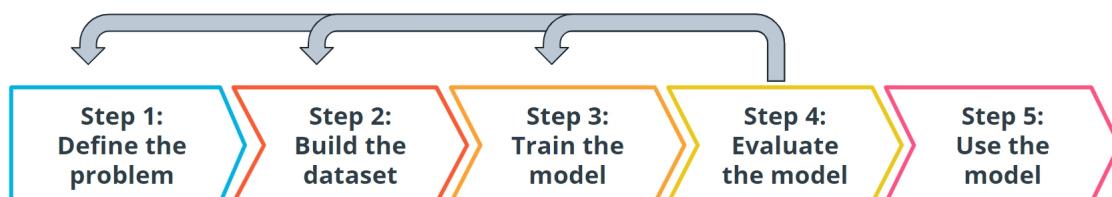


For example, let's say you're trying to predict how likely a customer is to buy either a jacket or t-shirt.

Log loss could be used to understand your model's uncertainty about a given prediction. In a single instance, your model could predict with 5% certainty that a customer is going to buy a t-shirt. In another instance, your model could predict with 80% certainty that a customer is going to buy a t-shirt. Log loss enables you to measure how strongly the model believes that its prediction is accurate.

In both cases, the model predicts that a customer will buy a t-shirt, but the model's certainty about that prediction can change.

Remember: This Process is Iterative



Every step we have gone through is highly iterative and can be changed or re-scoped during the course of a project. At each step, you might find that you need to go back and reevaluate some assumptions you had in previous steps. Don't worry! This ambiguity is normal.

Terminology

Log loss seeks to calculate how *uncertain* your model is about the predictions it is generating.

Model Accuracy is the fraction of predictions a model gets right.

Additional reading

The tools used for model evaluation are often tailored to a specific use case, so it's difficult to generalize rules for choosing them. The following articles provide use cases and examples of specific metrics in use.

1. [This healthcare-based example](#), which automates the prediction of spinal pathology conditions, demonstrates how important it is to avoid false positive and false negative predictions using the tree-based `xgboost` model.
2. The popular [open-source library](#) `sklearn` provides information about common metrics and how to use them.
3. [This entry from the AWS Machine Learning blog](#) demonstrates the importance of choosing the correct model evaluation metrics for making accurate energy consumption estimates using Amazon Forecast.

NEXT

Thinking deeper about linear regression

This lesson has covered linear regression in detail, explaining how you can envision minimizing loss, how the model can be used in various scenarios, and the importance of data.

What are some methods or tools that could be useful to consider when evaluating a linear regression output? Can you provide an example of a situation in which you would apply that method or tool?

Things to think about

There are many different tools that can be used to evaluate a linear regression model. Here are a few examples:

1. Mean absolute error (MAE): This is measured by taking the average of the absolute difference between the actual values and the predictions. Ideally, this difference is minimal.
- 2.
3. Root mean square error (RMSE): This is similar MAE, but takes a slightly modified approach so values with large error receive a higher penalty. RMSE takes the square root of the average squared difference between the prediction and the actual value.
- 4.

5.

Coefficient of determination or R-squared (R^2): This measures how well-observed outcomes are actually predicted by the model, based on the proportion of total variation of outcomes.

6.

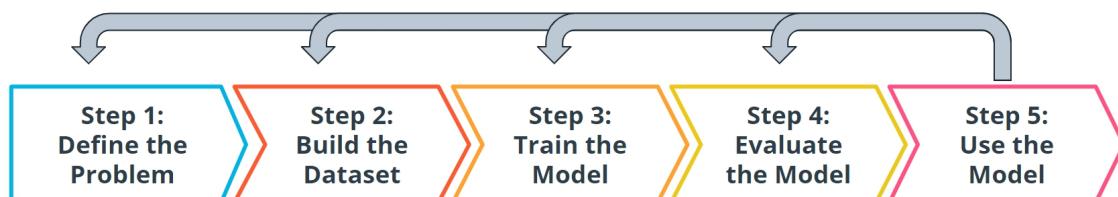
Step Five: Model Inference

Summary

Congratulations! You're ready to deploy your model.

Once you have trained your model, have evaluated its effectiveness, and are satisfied with the results, you're ready to generate predictions on real-world problems using unseen data in the field. In machine learning, this process is often called **inference**.

Iterative Process



Even after you deploy your model, you're always monitoring to make sure your model is producing the kinds of results that you expect. There may be times where you reinvestigate the data, modify some of the parameters in your model training algorithm, or even change the model type used for training.

NEXT

Introduction to Examples

Through the remainder of the lesson, we will be walking through 3 different case study examples of machine learning tasks actually solving problems in the real world.

- **Supervised learning**
Using machine learning to predict housing prices in a neighborhood based on lot size and number of bedrooms
- **Unsupervised learning**

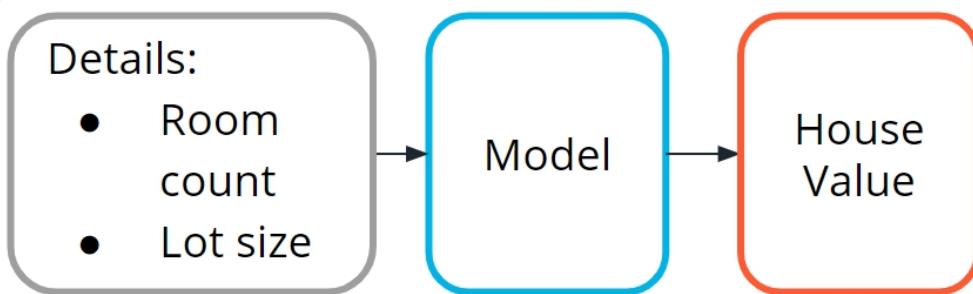
- Using machine learning to isolate micro-genres of books by analyzing the wording on the back cover description.
- **Deep neural network**
- While this type of task is beyond the scope of this lesson, we wanted to show you the power and versatility of modern machine learning. You will see how it can be used to analyze raw images from lab video footage from security cameras, trying to detect chemical spills.

Example One: House Price Prediction

House price prediction is one of the most common examples used to introduce machine learning.

Traditionally, real estate appraisers use many quantifiable details about a home (such as number of rooms, lot size, and year of construction) to help them estimate the value of a house.

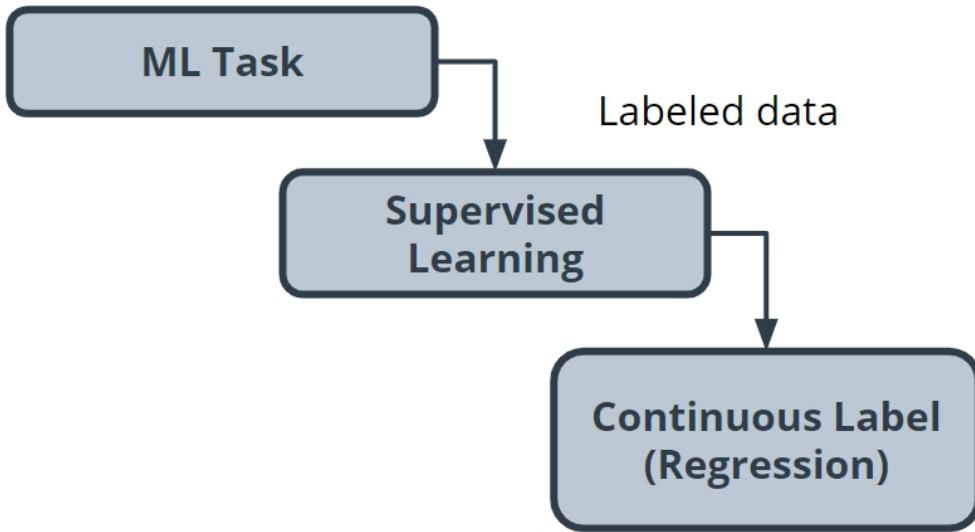
You detect this relationship and believe that you could use machine learning to predict home prices.



Step One: Define the Problem

Can we estimate the price of a house based on lot size or the number of bedrooms?

You access the sale prices for recently sold homes or have them appraised. Since you have this data, this is a *supervised learning* task. You want to predict a continuous numeric value, so this task is also a *regression* task.

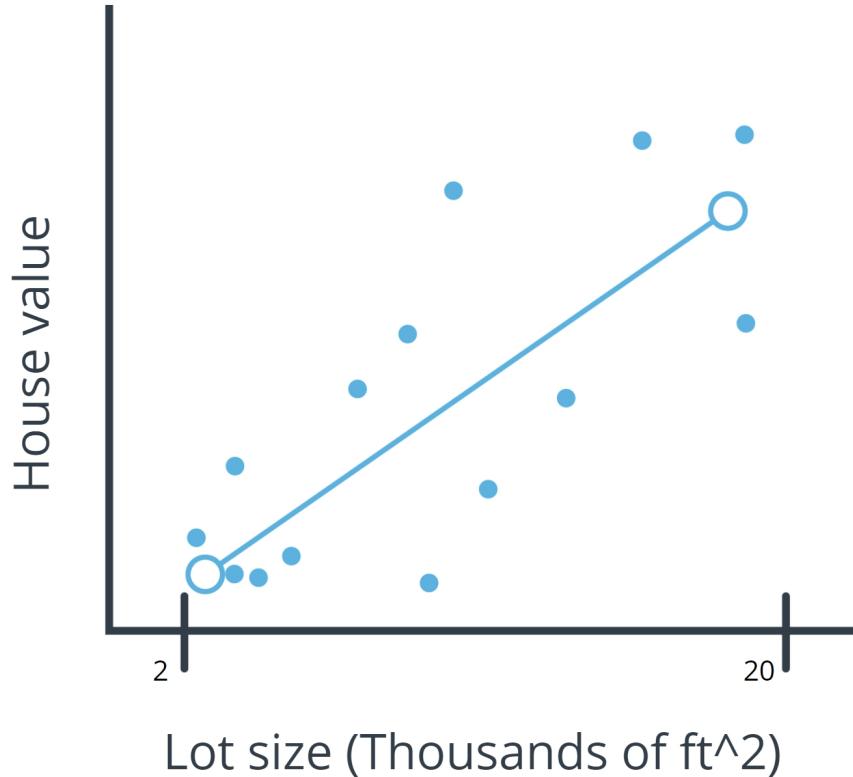


Step Two: Building a Dataset

- **Data collection:** You collect numerous examples of homes sold in your neighborhood within the past year, and pay a real estate appraiser to appraise the homes whose selling price is not known.
- **Data exploration:** You confirm that all of your data is numerical because most machine learning models operate on sequences of numbers. If there is textual data, you need to transform it into numbers. You'll see this in the next example.
- **Data cleaning:** Look for things such as missing information or outliers, such as the 10-room mansion. Several techniques can be used to handle outliers, but you can also just remove those from your dataset.

| # of Rooms | Lot Size (ft ²) | House Value (\$) |
|------------|-----------------------------|------------------|
| 4 | 10,454 | 339,900 |
| 3 | 9,147 | 239,000 |
| 3 | 10,890 | 250,000 |
| 10 | 25,877 | 877,000 |

- **Data visualization:** You can plot home values against each of your input variables to look for trends in your data. In the following chart, you see that when lot size increases, the house value increases.



Step Three: Model Training

Prior to actually training your model, you need to split your data. The standard practice is to put 80% of your dataset into a training dataset and 20% into a test dataset.

Linear model selection

As you see in the preceding chart, when lot size increases, home values increase too. This relationship is simple enough that a linear model can be used to represent this relationship.

A linear model across a single input variable can be represented as a line. It becomes a plane for two variables, and then a hyperplane for more than two variables. The intuition, as a line with a constant slope, doesn't change.

Using a Python library

The Python `scikit-learn` library has tools that can handle the implementation of the model training algorithm for you.

Step Four: Evaluation

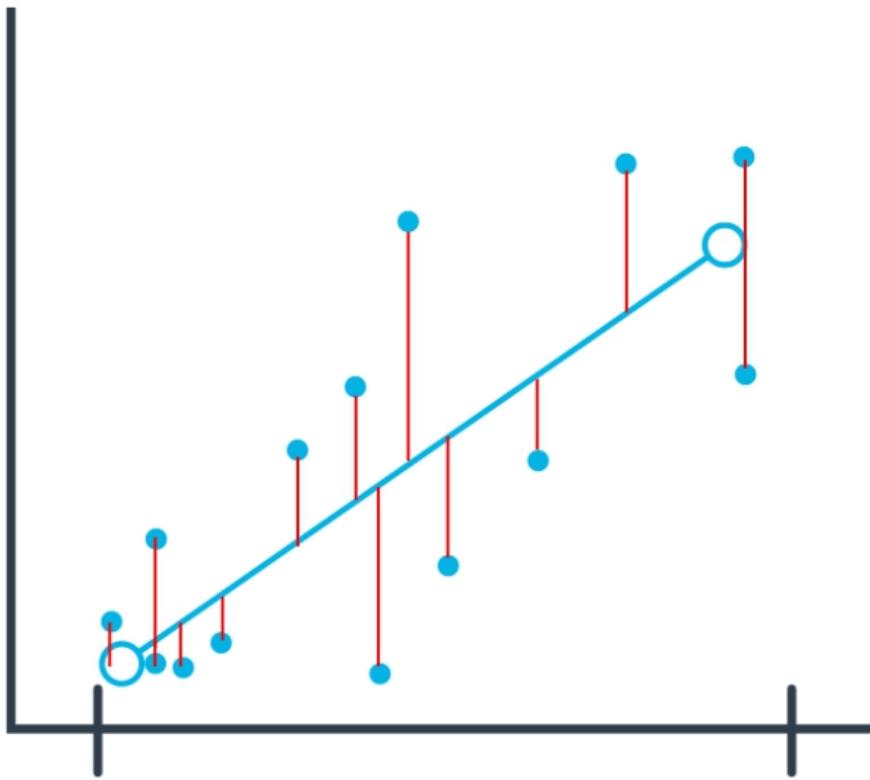
One of the most common evaluation metrics in a regression scenario is called *root mean square* or *RMS*. The math is beyond the scope of this lesson, but RMS can be thought of roughly as the "average error" across your test dataset, so you want this value to be low.

$$RMS = \sqrt{\frac{1}{n} \sum_i x_i^2}$$

The math behind RMS

In the following chart, you can see where the data points are in relation to the blue line. You want the data points to be as close to the "average" line as possible, which would mean less net error.

You compute the *root mean square* between your model's prediction for a data point in your test dataset and the true value from your data. This actual calculation is beyond the scope of this lesson, but it's good to understand the process at a high level.



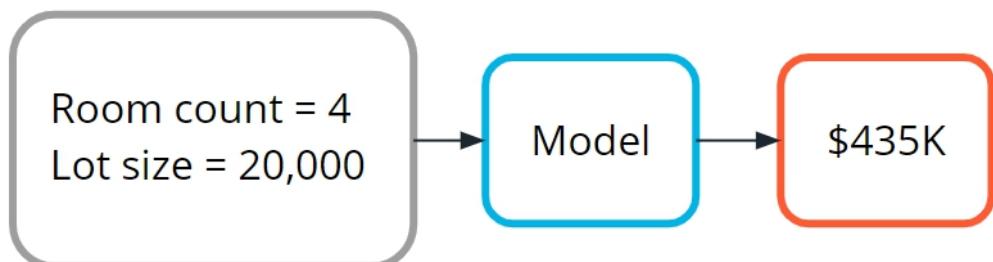
Interpreting Results

In general, as your model improves, you see a better RMS result. You may still not be confident about whether the specific value you've computed is good or bad.

Many machine learning engineers manually count how many predictions were off by a threshold (for example, \$50,000 in this house pricing problem) to help determine and verify the model's accuracy.

Step Five: Inference: Try out your model

Now you are ready to put your model into action. As you can see in the following image, this means seeing how well it predicts with new data not seen during model training.



Terminology

- **Continuous:** Floating-point values with an infinite range of possible values. The opposite of categorical or discrete values, which take on a limited number of possible values.
- **Hyperplane:** A mathematical term for a surface that contains more than two planes.
- **Plane:** A mathematical term for a flat surface (like a piece of paper) on which two points can be joined by a straight line.
- **Regression:** A common task in supervised machine learning.

Additional reading

The [Machine Learning Mastery](#) blog is a fantastic resource for learning more about machine learning. The following example blog posts dive deeper into training regression-based machine learning models.

- [How to Develop Ridge Regression Models in Python](#) offers another approach to solving the problem in the example from this lesson.
- Regression is a popular machine learning task, and you can use [several different model evaluation metrics](#) with it.

Challenge yourself

In this example, we used a linear model to solve a simple regression supervised learning task. This model type is a great first choice when exploring a machine learning problem because it's very fast and straightforward to train. It typically works well when you have relationships in your data that are linear (when input changes by X, output changes by some fixed multiple of X).

Can you think of an example of a problem that would not be solvable by a linear model?

Your reflection

To guess the title of the book by the reader

Things to think about

Linear models typically fail when there is no helpful linear relationship between the input variables and the label.

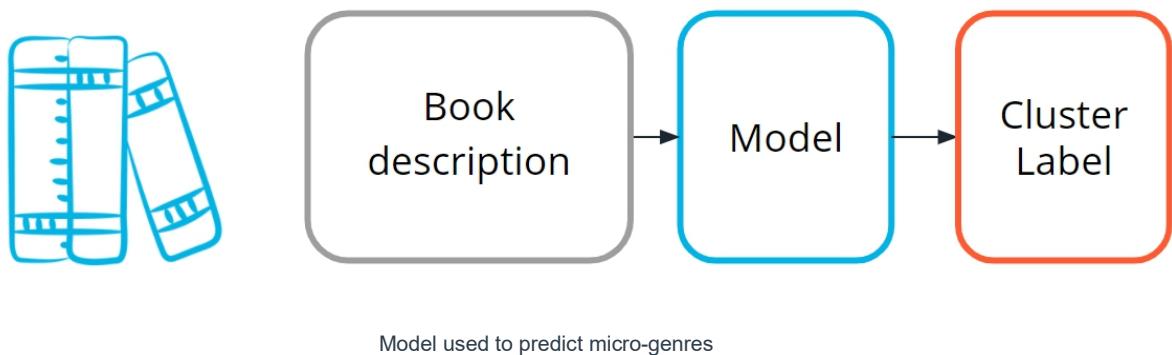
For example, imagine predicting the height (label) of a thrown projectile over time (input variable). You know the trajectory is not linear; it's curved. Any straight line you try to use to describe this phenomenon would be invalid for a large range of the projectile's trajectory.

Techniques do exist to modify your data so you can still use linear models in these situations. Such methods are out of scope for this course but are called *kernel methods*.

Example Two: Book Genre Exploration

In this video, you saw how the machine learning process can be applied to an unsupervised machine learning task that uses book description text to identify different micro-genres.

Step One: Define the Problem

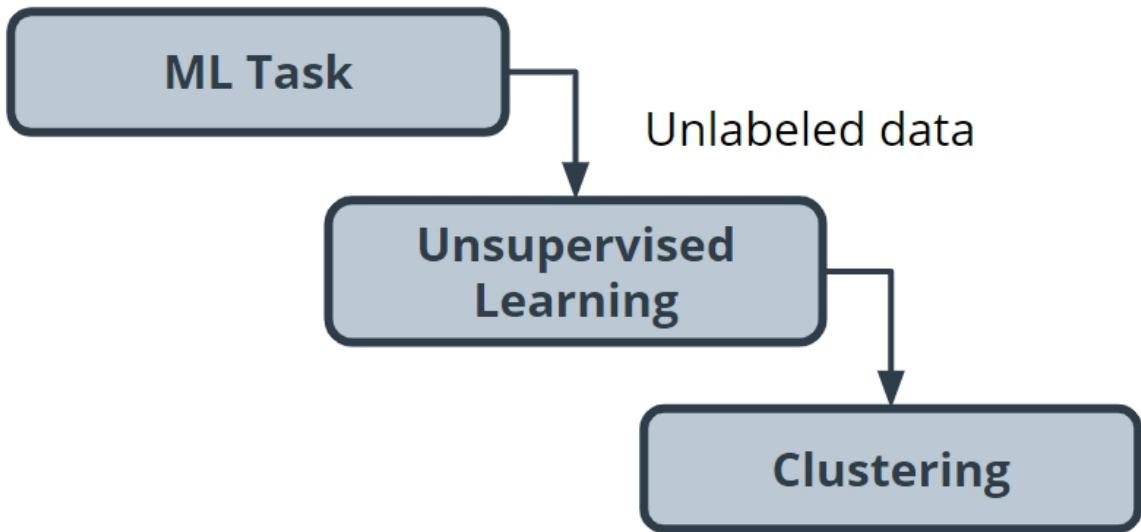


Find clusters of similar books based on the presence of common words in the book descriptions.

You do editorial work for a book recommendation company, and you want to write an article on the largest book trends of the year. You believe that a trend called "micro-genres" exists, and you have confidence that you can use the book description text to identify these micro-genres.

By using an unsupervised machine learning technique called *clustering*, you can test your hypothesis that the book description text can be used to identify these "hidden" micro-genres.

Earlier in this lesson, you were introduced to the idea of unsupervised learning. This machine learning task is especially useful when your data is not labeled.



Unsupervised learning using clustering

Step Two: Build your Dataset

To test the hypothesis, you gather book description text for 800 romance books published in the current year.

Data exploration, cleaning and preprocessing

For this project, you believe capitalization and verb tense will not matter, and therefore you remove capitals and convert all verbs to the same tense using a Python library built for processing human language. You also remove punctuation and words you don't think have useful meaning, like 'a' and 'the'. The machine learning community refers to these words as *stop words*.

Before you can train the model, you need to do some data preprocessing, called *data vectorization*, to convert text into numbers.

You transform this book description text into what is called a *bag of words* representation shown in the following image so that it is understandable by machine learning models.

How the bag of words representation works is beyond the scope of this course. If you are interested in learning more, see the **Additional Reading** section at the bottom of the page.

"Little did he know, she was secretly a vampire."



[‘little’, ‘does’, ‘he’, ‘know’, ‘she’, ‘is’, ‘secretly’, ‘vampire’]



Bag of Words



[0, 0, 1, 0, 1, ...]

Step Three: Train the Model

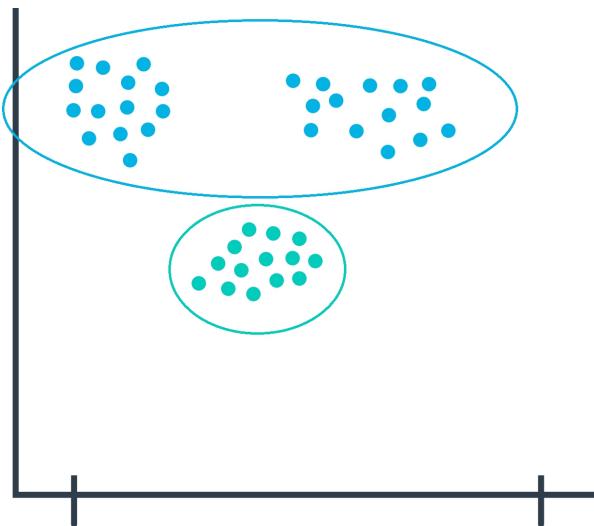
Now you are ready to train your model.

You pick a common cluster-finding model called **k-means**. In this model, you can change a model parameter, **k**, to be equal to how many clusters the model will try to find in your dataset.

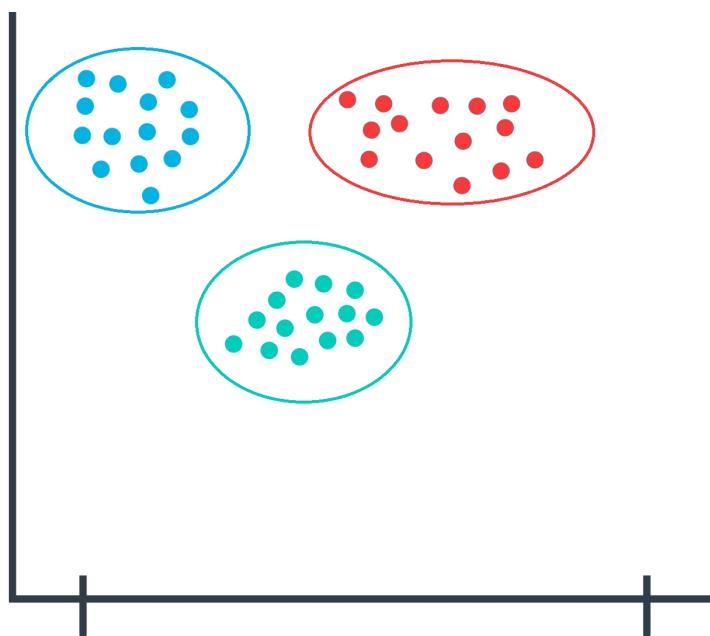
Your data is unlabeled: you don't know how many microgenres might exist. So you train your model multiple times using different values for **k** each time.

What does this even mean? In the following graphs, you can see examples of when **k=2** and when **k=3**.

K=2



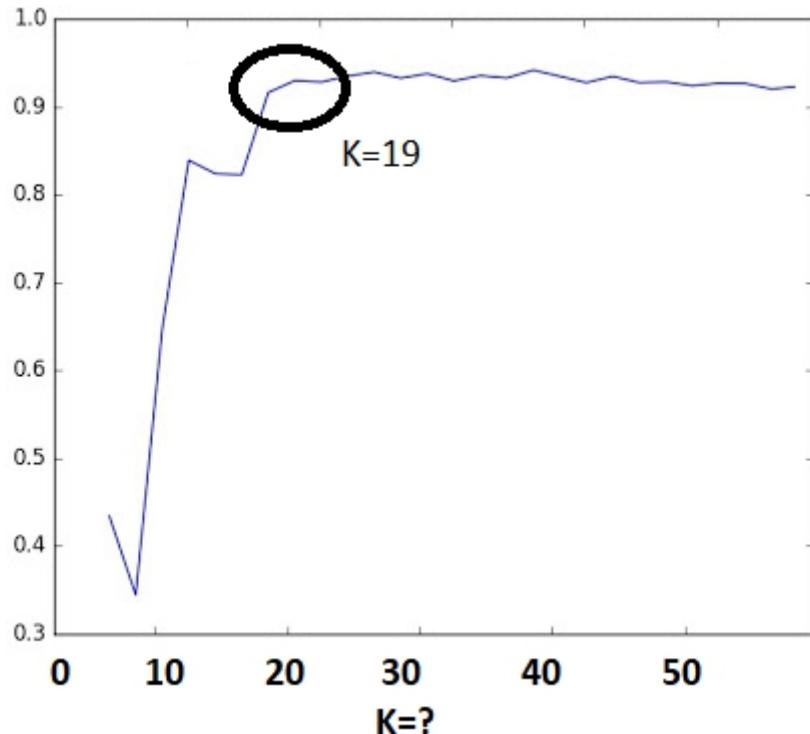
K=3



During the model evaluation phase, you plan on using a metric to find which value for k is most appropriate.

Step Four: Model Evaluation

In machine learning, numerous statistical metrics or methods are available to evaluate a model. In this use case, the *silhouette coefficient* is a good choice. This metric describes how well your data was clustered by the model. To find the optimal number of clusters, you plot the silhouette coefficient as shown in the following image below. You find the optimal value is when $k=19$.



Optimum number ($k=19$) of clusters

Often, machine learning practitioners do a manual evaluation of the model's findings.

You find one cluster that contains a large collection of books you can categorize as "paranormal teen romance." This trend is known in your industry, and therefore you feel somewhat confident in your machine learning approach. You don't know if every cluster is going to be as cohesive as this, but you decide to use this model to see if you can find anything interesting about which to write an article.

Step Five: Inference (Use the Model)

As you inspect the different clusters found when $k=19$, you find a surprisingly large cluster of books. Here's an example from fictionalized cluster #7.

| Cluster Label | Book Description |
|---------------|---|
| 7 | “Susan’s crush just moved away..” |
| 7 | “Can Alice and Bob keep their relationship together three hundred miles apart?” |
| 7 | “When Hank’s fiance George got offered a new job in New York...” |

Clustered data

As you inspect the preceding table, you can see that most of these text snippets are indicating that the characters are in some kind of long-distance relationship. You see a few other self-consistent clusters and feel you now have enough useful data to begin writing an article on unexpected modern romance microgenres.

Terminology

- **Bag of words:** A technique used to extract features from the text. It counts how many times a word appears in a document (corpus), and then transforms that information into a dataset.
- **Data vectorization:** A process that converts non-numeric data into a numerical format so that it can be used by a machine learning model.
- **Silhouette coefficient:** A score from -1 to 1 describing the clusters found during modeling. A score near zero indicates overlapping clusters, and scores less than zero indicate data points assigned to incorrect clusters. A score approaching 1 indicates successful identification of discrete non-overlapping clusters.
- **Stop words:** A list of words removed by natural language processing tools when building your dataset. There is no single universal list of stop words used by all-natural language processing tools.

Additional reading

[Machine Learning Mastery](#) is a great resource for finding examples of machine learning projects.

- The [How to Develop a Deep Learning Bag-of-Words Model for Sentiment Analysis \(Text Classification\)](#) blog post provides an example using a bag of words-based approach pair with a deep learning model.

Example Three: Spill Detection from Video

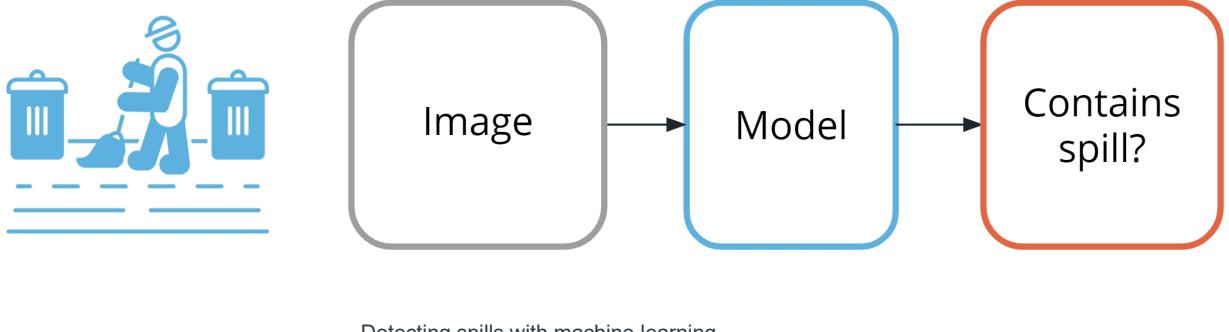
In the previous two examples, we used classical methods like linear models and k-means to solve machine learning tasks. In this example, we'll use a more modern model type.

Note: *This example uses a neural network. The algorithm for how a neural network works is beyond the scope of this lesson. However, there is still value in seeing how machine learning applies in this case.*

Step One: Defining the Problem

Imagine you run a company that offers specialized on-site janitorial services. A client, an industrial chemical plant, requires a fast response for spills and other health hazards. You realize if you could automatically detect spills using the plant's surveillance system, you could mobilize your janitorial team faster.

Machine learning could be a valuable tool to solve this problem.

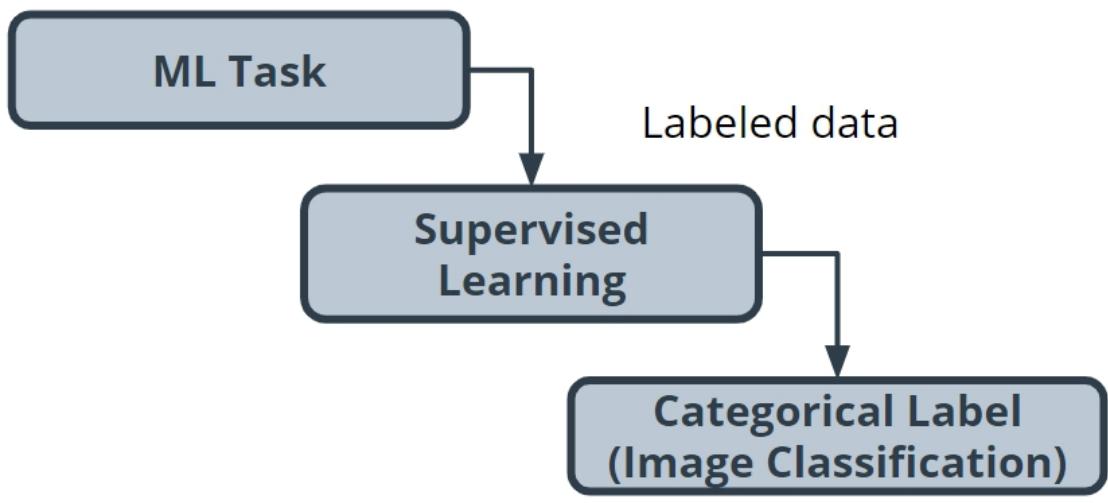


Detecting spills with machine learning

Step Two: Model Training (and selection)

This task is a supervised classification task, as shown in the following image. As shown in the image above, your goal will be to predict if each image belongs to one of the following classes:

- Contains spill
- Does not contain spill



Step Two: Building a Dataset

- **Collecting**
Using historical data, as well as safely staged spills, you quickly build a collection of images that contain both spills and non-spills in multiple lighting conditions and environments.
 - **Exploring and cleaning**
You go through all the photos to ensure the spill is clearly in the shot. There are Python tools and other techniques available to improve image quality, which you can use later if you determine a need to iterate.
 - **Data vectorization** (converting to numbers)
Many models require numerical data, so all your image data needs to be transformed into a numerical format. Python tools can help you do this automatically.
 - In the following image, you can see how each pixel in the image on the left can be represented in the image on the right by a number between 0 and 1, with 0 being completely black and 1 being completely white.



Chemical spill image

Numeric representation of chemical spill image

Split the data

- You split your image data into a training dataset and a test dataset.

Step Three: Model Training

Traditionally, solving this problem would require hand-engineering features on top of the underlying pixels (for example, locations of prominent edges and corners in the image), and then training a model on these features.

Today, deep neural networks are the most common tool used for solving this kind of problem. Many deep neural network models are structured to learn the features on top of the underlying pixels so you don't have to learn them. You'll have a chance to take a deeper look at this in the next lesson, so we'll keep things high-level for now.

CNN (convolutional neural network)

Neural networks are beyond the scope of this lesson, but you can think of them as a collection of very simple models connected together. These simple models are called *neurons*, and the connections between these models are trainable model parameters called *weights*.

Convolutional neural networks are a special type of neural network particularly good at processing images.

Step Four: Model Evaluation

As you saw in the last example, there are many different statistical metrics you can use to evaluate your model. As you gain more experience in machine learning, you will learn how to research which metrics can help you evaluate your model most effectively. Here's a list of common metrics:

| | | |
|------------------|---------------------------|-------------|
| Accuracy | False positive rate | Precision |
| Confusion matrix | False negative rate | Recall |
| F1 Score | Log Loss | ROC curve |
| | Negative predictive value | Specificity |

In cases such as this, accuracy might not be the best evaluation mechanism.

Why not? You realize the model will see the '**Does not contain spill**' class almost all the time, so any model that just predicts "**no spill**" most of the time will seem pretty accurate.

What you really care about is an evaluation tool that rarely misses a real spill.

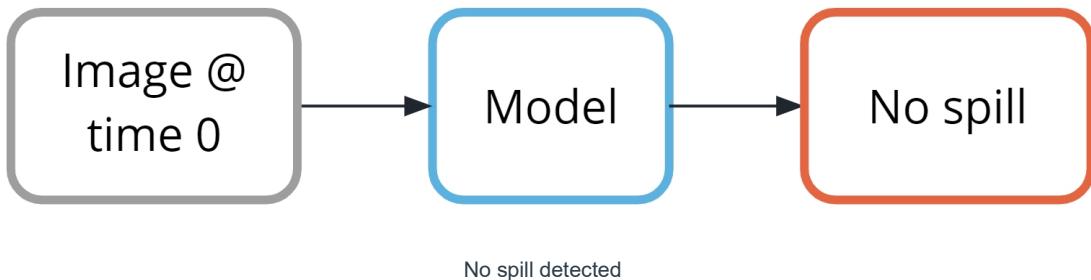
After doing some internet sleuthing, you realize this is a common problem and that **Precision** and **Recall** will be effective. You can think of *precision* as answering the question, "Of all predictions of a spill, how many were right?" and *recall* as answering the question, "Of all actual spills, how many did we detect?"

Manual evaluation plays an important role. You are unsure if your staged spills are sufficiently realistic compared to actual spills. To get a better sense how well your model performs with actual spills, you find additional examples from historical records. This allows you to confirm that your model is performing satisfactorily.

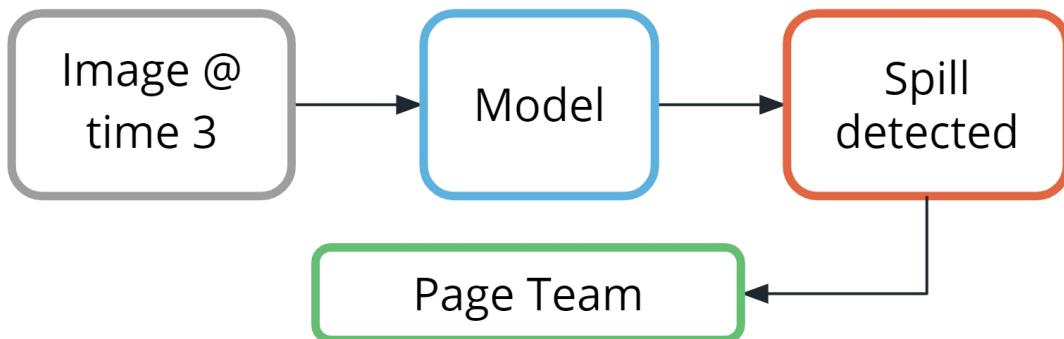
Step Five: Model Inference

The model can be deployed on a system that enables you to run machine learning workloads such as AWS Panorama.

Thankfully, most of the time, the results will be from the class '**Does not contain spill**'.



But, when the class '**Contains spill**' is detected, a simple paging system could alert the team to respond.



Terminology

Convolutional neural networks(CNN) are a special type of neural network particularly good at processing images.

Neural networks: a collection of very simple models connected together.

- These simple models are called **neurons**
- the connections between these models are trainable model parameters called **weights**.

Additional reading

As you continue your machine learning journey, you will start to recognize problems that are excellent candidates for machine learning.

The [AWS Machine Learning Blog](#) is a great resource for finding more examples of machine learning projects.

- In the [Protecting people from hazardous areas through virtual boundaries with Computer Vision](#) blog post, you can see a more detailed example of the deep learning process described in this lesson.

Lesson Review

Congratulations on making it through the lesson. Let's review what you learned

- In the first part of the lesson, we talked about what machine learning actually is, introduced you to some of the most common terms and ideas used in machine learning, and identified the common components involved in machine learning projects.
- We learned that machine learning involves using trained models to generate predictions and detect patterns from data. We looked behind the scenes to see what is really happening. We also broke down the different steps or tasks involved in machine learning.
- We looked at three machine learning examples to demonstrate how each works to solve real-world situations.
- A **supervised learning** task in which you used machine learning to predict housing prices for homes in your neighborhood, based on the lot size and the number of bedrooms.
- An **unsupervised learning** task in which you used machine learning to find interesting collections of books in a book dataset, based on the descriptive words in the book description text.
- Using a **deep neural network** to detect chemical spills in a lab from video and images.