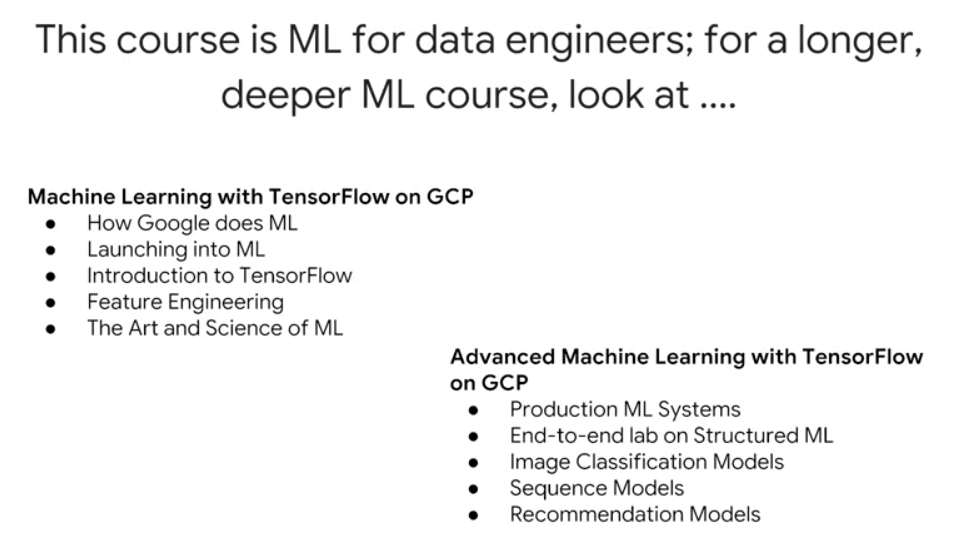
Serverless Machine Learning on Google Cloud Platform

Machine learning is a way to derive insight from data. We look at what machine learning is, how to build machine learning programs, how to think about machine learning programs, and also how to write transfer flow programs. We will step through transfer flow so that you can basically build your own transfer flow programs and execute them again in a cluster-less manner, fully managed on Google Cloud. We'll look at how to do hyperparameter tuning to improve your models, how to do feature engineering to create better features - again to improve your models.



Machine Learning with TensorFlow on GCP: <https://www.coursera.org/specializations/machine-learning-tensorflow-gcp>

Advanced Machine Learning with TensorFlow on GCP: This course is apparently no longer offered

Deep Learning (DeepLearning.ai) – much more math-oriented: <https://www.coursera.org/specializations/deep-learning>

Fast.ai – highly recommended by several colleagues as being more hands-on and less math oriented: <http://course.fast.ai/>

## How to think about Machine Learning

Machine learning is a very general function. For example, with a neural network, you determine weights on that function which minimize some cost function. These are configurable parameters, these are tunable parameters. Parameters of the function are “tuned” based on a known dataset that represents a greater dataset. Then, apply this function that you have tuned on this known dataset and you use to predict unknown values.

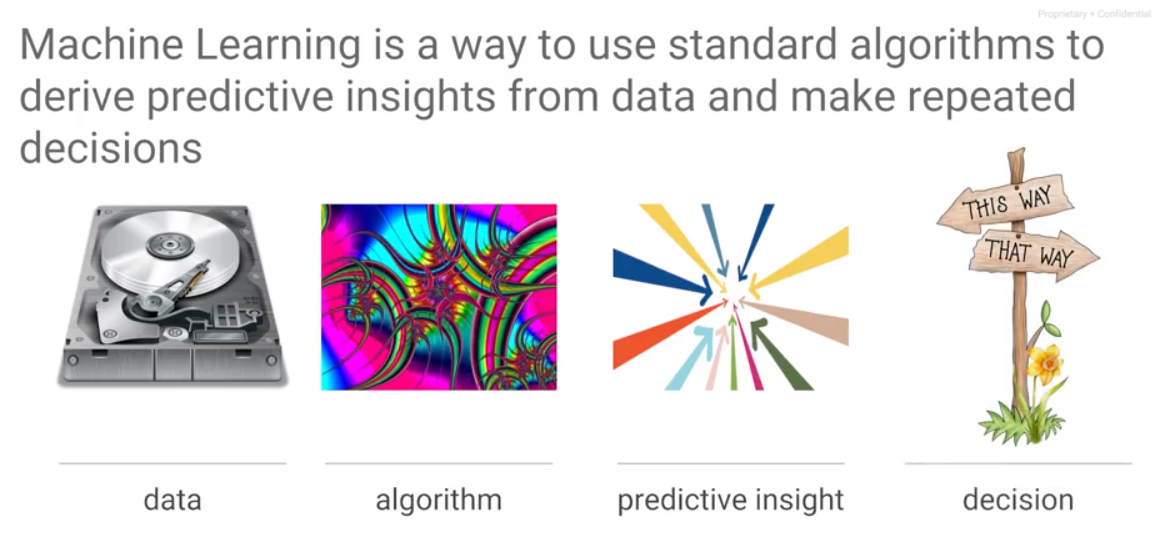
Machine learning, or the approximation of this function is typically performed on a large amount of data, and it needs to be done scale. And that's something that Google Cloud does very well. What this class presents is machine learning on Google Cloud platform using TensorFlow and using Cloud ML. But you will also realize that with Cloud ML, you get serverless machine learning, which essentially means that the amount of configuration and work that you need to do is much reduced. You don't have to manage a cluster of machines, and you basically write TensorFlow code and you submit it to the cloud, and Cloud ML essentially takes care of whole of running it for you.

In this course you are introduced to the basics of machine learning. Then we'll delve into how you can apply machine learning to solve a particular problem.

# Module 1: Getting started with machine learning

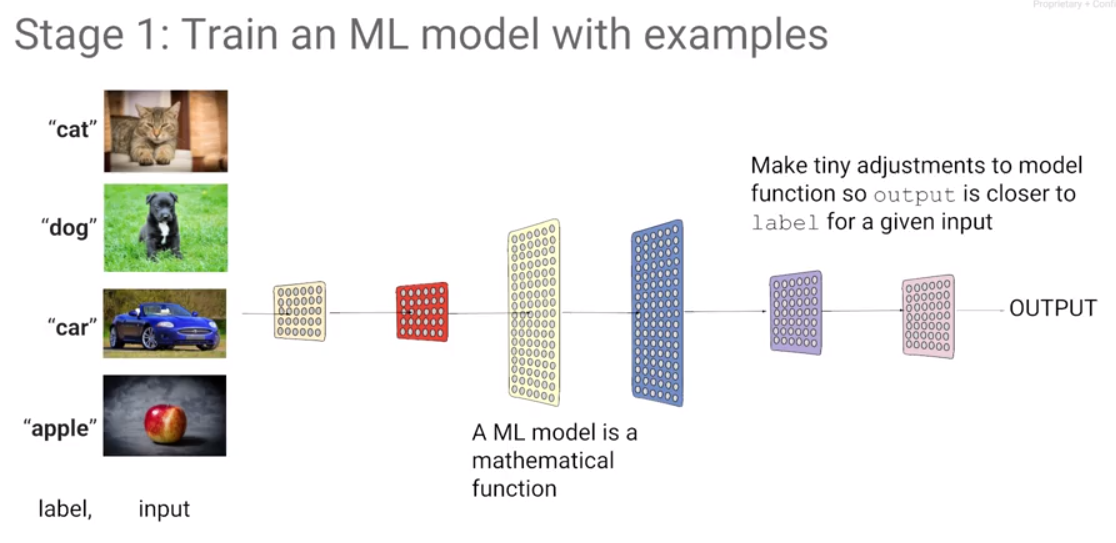
## What is Machine Learning (ML)?

In this chapter, we define machine learning. We look at machine learning from the point of view of playing with it. That's one of the best ways to learn what something is. And then we will look at how to create effective machine learning models. And finally, we will create the machine learning datasets that we will use in the rest of this module.

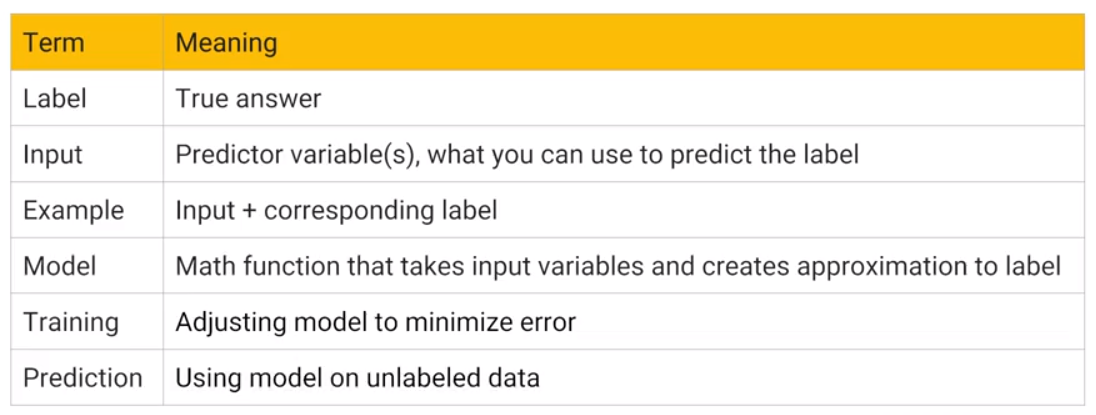
 Machine learning is a way to derive insights from data. Given a large set of data and standard algorithms, you can use to obtain insight from the data. The kinds of insights that you would get from the data tend to be predictive in nature. That's the way I distinguish between things like business intelligence, which is all about historical data, trying to figure out what happened, and machine learning, which is about training the machine learning model on older data – that is true – but to be able to apply that model to unknown data, to be able to predict with it.

This concept of ML always being for image recognition is limiting, however useful it is in giving an example. ML can do far more than just recognize images – it needs to be generalized at least to the level of pattern recognition.

When you think about machine learning, you basically think in terms of what you want to accomplish with it. Let's say that what we want to accomplish is to take bunch of images and determine what's in those images. For example, for this image, is the animal a cat or dog. Or for the third image is the image a car or the output on the fourth image needs to be an apple. In order to do that, we need examples (supervised learning). An example in machine learning terms is a combination of the input, the input for which we want an output, and a label, which is a true output, the thing that we know, this is what it needs to be. For example, we have an image the label cat and the second image the label dog and the third image the label car. The pair of label and input together form an example. When training a machine learning model, we train it with examples which are combinations of labels and inputs. Once you have those examples, the machine learning model is a mathematical function that is trained. The way you train the model is that any of the mathematical models have free parameters, tunable parameters called weights, and you adjust those weights in such a way that the output of the ML model of this function matches the first image which is a cat and the second image which is a dog. If we had trained this model such that given this image the label were to be grass, that is what the ML model would learn. What the machine learning model does is lern the labels for a particular image. It determines a function such that that function given this input or given any of these inputs is going to match corresponding label. Given such a function, we can now give it a new image, an image for which we don't know the label, and the resulting function would give you a prediction, and that prediction is going to be the right output for this image.



The whole idea of machine learning is that given a large data set of labeled data, adjust the mathematical model in such a way that given an input, the output for that input matches the original label. Then you don't need the training data anymore. All you need is a model, and you can take that model and apply it to an arbitrary image and hopefully what you get back is what you have trained that model to do on an image like that image. Predict with a model that has been trained.



A label is the correct output for some input. This is what you train the model with. The label is a correct output for an input.

The input is the thing that you will know and that you can provide at the time of even prediction. These are the things, for example, if they're images; the image itself is an input.

An example is a combination of label and input. An input and its corresponding label together form an example.

A model is this mathematical function that takes an input and creates an output that approximates the label for that input.

Training is this process of adjusting the weights of a model in such a way that it can make predictions, given an input.

Prediction is this process of taking an input in and applying the mathematical model to it, so as to get an output that hopefully is the correct output for that input.

Supervised learning is learning by example. There's another type of machine learning use case, where you don't have labels. That’s called clustering. For example, you may have the data, as the data on the left, you may have some data on people. Every dot here is a person at a company. We know the number of years they have spent at the company and we know their income, and we can look at this data and we feel that this data falls into two big categories: one category of people whose income rises much faster than this other category of people. And we may say anybody in this grouping is on the fast track at this company. But that's not a label, because we don't know the truth. This is just looking at the data and trying to divide it into two classes. This is the last time that we are going to look at unsupervised learning. The rest of this course, we are going to focus on supervised learning.

A very common source of structured data for machine learning is a data warehouse, e.g., BigQuery. We can do a select statement in BigQuery to create this data set such that we can use it to train a model. This is a very common use case - this is a use case that we are going to focus on in this module. We are going to be looking at structured data prediction for machine learning. You have a trained model, how does prediction work in a structured data problem?

The simplest model that is capable of doing something like this may have exactly just one layer, and that layer is a layer of weights. We add b, a constant that we will call the bias. This may be the entire mathematical model and you can see that we could probably adjust these weights in such a way that for any of these rows in the data, we can achieve the desired output. We can choose a regression model or a classification model.

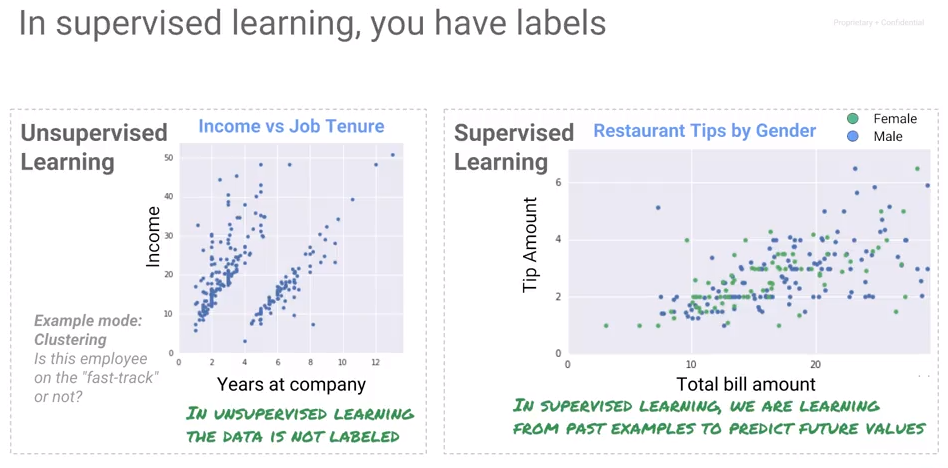
Given this input and this label, our training data set, we take our mathematical model and change the weights of this mathematical model in such a way that given any of these inputs, the output is as close as possible to the label that we have. For classification, a case is true or false (represent as one and zero), make a computation, a weighted sum, and that's our mathematical model. How does it work for unstructured data? How can we do a weighted sum of an image? Even an image, you can think of as a two-dimensional array of pixels and each pixel has red, green, blue and alpha, so four numbers. If you have an image, an image is a tensor.

What about if you have text? How does the word still become a number? Now that's a little bit harder. Take any word and map it to be a vector, e.g., the word "the" can be represented by the numbers [0 4 5 0 3 4]. We could assign arbitrary numbers to every word in the dictionary, but that is not ideal. It is not ideal because you would like to say that the word, the vector representation of plenty and the vector representation of many and the vector representation of much should all be relatively close to each other and should be very different from the vector representation of seeing because the word "seeing" and "plenty" are not as closely related as the word "much" and "plenty". This itself turns out to be a machine learning problem of assigning an appropriate vector to appropriate words in a language, but fortunately that's been done before. For a machine learning problem with freeform text as input, use word2vec, one of these techniques to convert words to vectors. Typically, there is a different word2vec for different languages, so you would go ahead and pick one of those word2vec objects and use it to convert your text into vectors. From there, you have numbers and use those as input to the machine learning problem.

Even though we are focusing primarily on structured data in this course, the treatment of other objects such as images and text uses the same concepts. As it turns out, the kind of layers and the kinds of things that you would do would be somewhat different, but we're not going to talk about those, we're going to stick very much to the basic concepts of machine learning and applying it to structured data machine learning, which is the most common kind of data problem that you would face in most businesses. To learn ML, learn it on structured data because it tends to be the most useful.

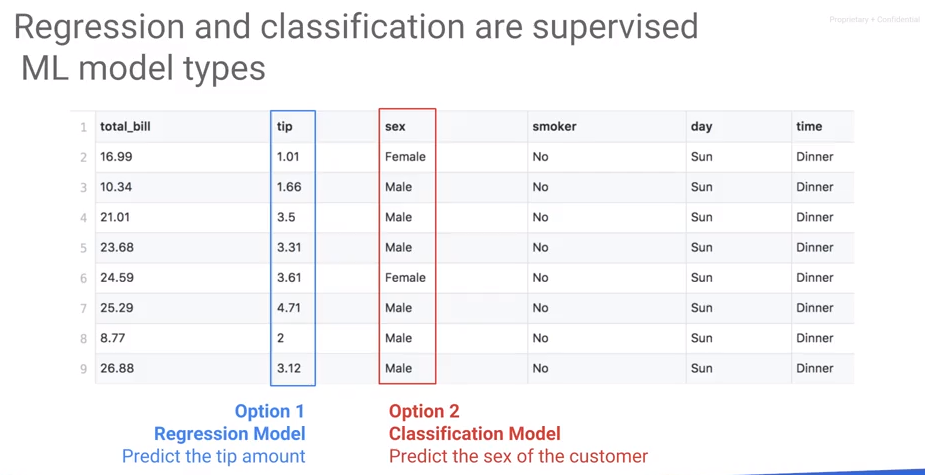
If you have a classification model, the labels are Boolean values (0 or 1) but the output of the model will be a number between zero and one. How do we interpret that number? We interpret that number as a probability such that if this probability is being one or zero. Given this framing, for a labeled data set with a known output for each of those inputs, we can now create a machine learning model, a mathematical function that will be able to create an approximation of that label for every one of those inputs.

## Types of ML

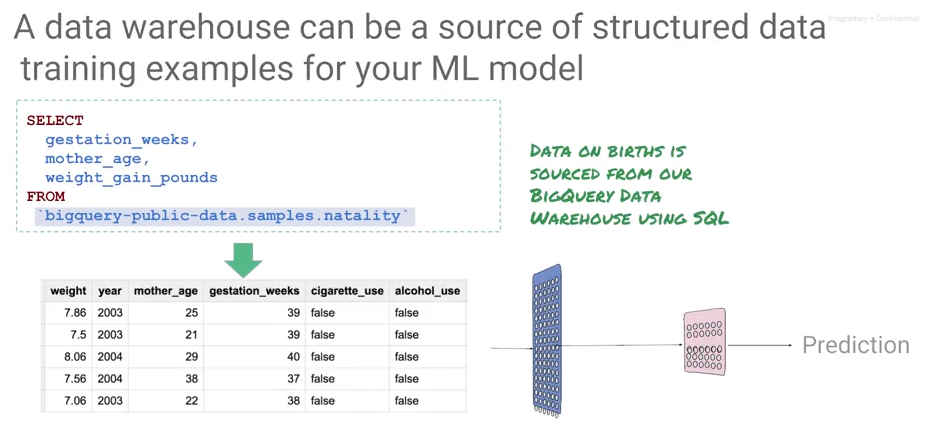


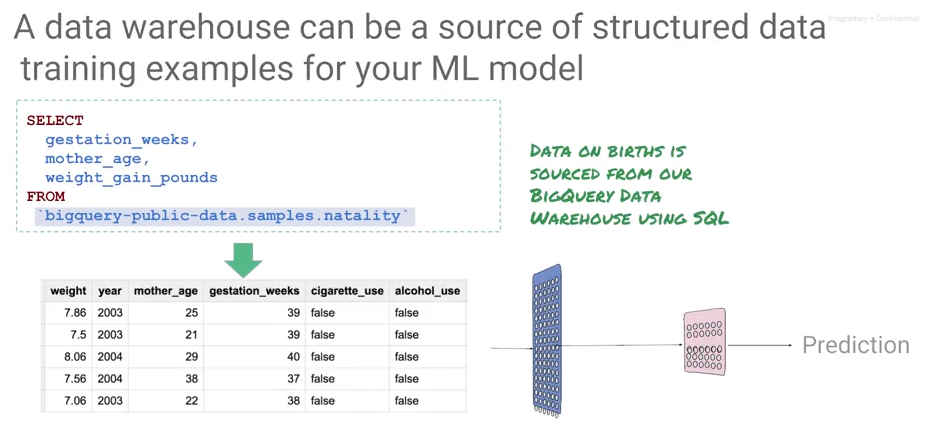
Supervised learning is learning by example. Another type of machine learning does not have labels. That's called clustering. This is just looking at the data and trying somewhat arbitrarily dividing it into different groups.

In this course, we concentrate on supervised learning where we have a label and we know the true outcome for some input. If the problem is to calculate an amount that is a continuous number, it is a regression problem. On the other hand, if the problem is to assign a categorical value out of a discrete (usually two – e.g., True and False) set, then it is a classification problem.

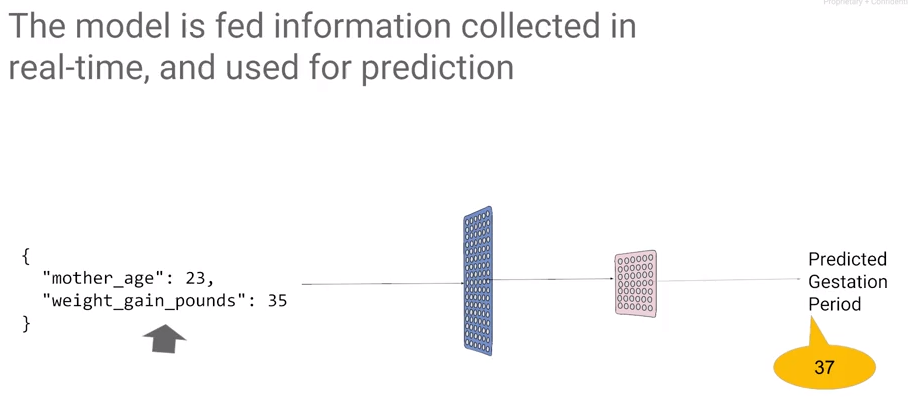


## The ML Pipeline

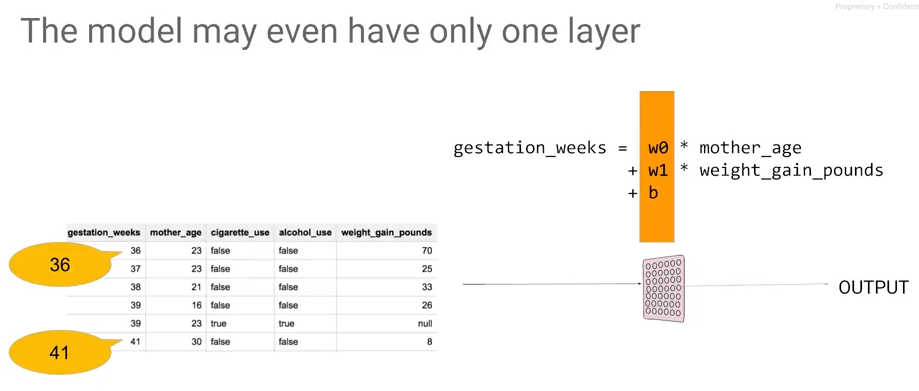




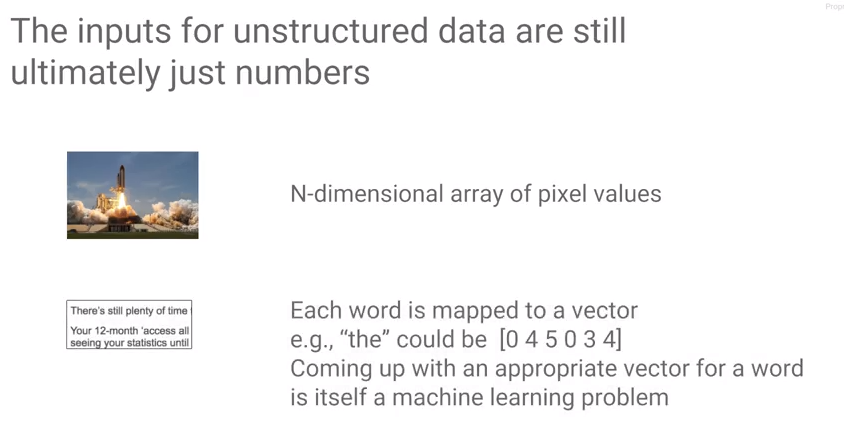
Here's another example of a regression model that predicts a continuous number. This is a dataset on babies that were born. And we know the mother's age, and the weight gain in the pregnancy so far. And based on this data set, we want to predict how long the gestation period is going to be? How long is a pregnancy going to last for this model? Now, where does this data come from? This is what we would call structured data. And a very common source of structure data for machine learning is your data warehouse - in this case, BigQuery. We can go ahead and do a select statement and BigQuery to create this data set such that, we can use it a train model that predicts the gestation weeks, given all of these attributes of the pregnancy. This is a very common use case and this is a use case that we want to focus on in this module. What are going to be doing, is we're going to be looking at structure data prediction for machine learning. Because what we're trying to predict is the gestation weeks and that is a continuous number, this model that takes the mother's age, the weight gain in pounds, the model that takes these two inputs, and uses it to predict the gestation weeks, this model is a regression model. Having created a data set, we next train a regression model based on data, in our data warehouse. Once you've trained a model how does prediction work in a structured data problem. Well, remember that you trained a model on two inputs; mother's age and weight gain in pounds. If you want to do a prediction, you need to have the person who requires a prediction, the mother. Or it's usually computer's application making the prediction on behalf of this patient. That application needs to give you those two input parameters. And the machine learning service will give you back the predicted gestation weeks. In other words, the inputs are going to be the mother's age, and the weight gain in pounds. And I'm showing it to you here as json input. This is presented to the machine learning model, which applies the mathematical function and outcomes the gestation period, because that's what the model was trained to predict.



## Variants of an ML Model

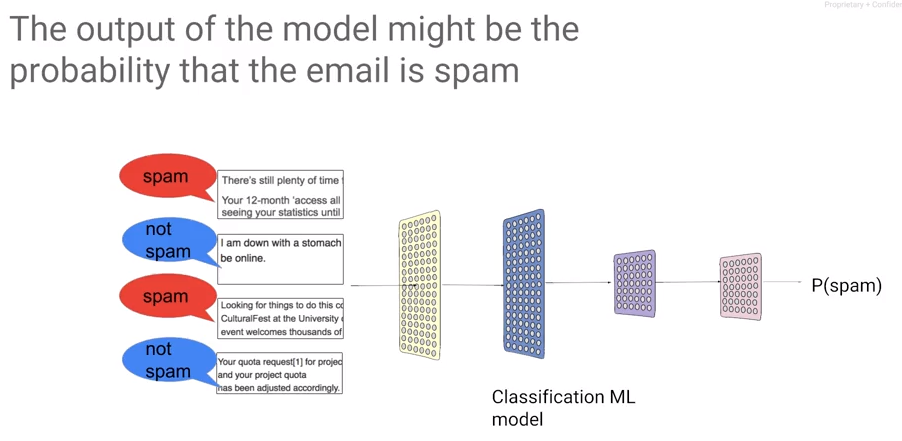


The simplest model is one layer of weights. We can write the mathematical function as gestation weeks is weight zero times the mother's age, plus weight 1 times the weight gain in pounds plus B, a constant that we will call the bias. This may be the entire mathematical model, and you can see that we could probably adjust these weights in such a way that for any of the rows in the data, the gestation weeks that we get by applying this formula to the two columns, is as close as possible to the gestation week for that role. The other type of model besides the regression model is a classification model. For example, given the text of an e-mail and the label, predict if it is spam or not.

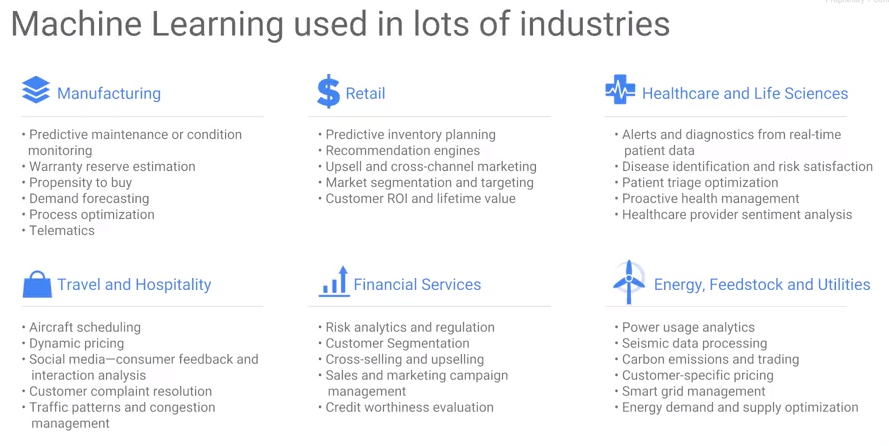
 How does this work if the inputs are images or text? Everything that goes into a machine learning model has to be numeric. But how does it work for unstructured data? How can we do a weighted sum of an image? An image may be represented as a two-dimensional array of pixels and each pixel has red, green, blue and alphas for numbers. An image is a three-dimensional array of numbers. A 1D array is a vector, a 2D array is a matrix, a three dimensional - in general we call an n-dimensional array a tensor. That is where the name TensorFlow comes from. An image is a tensor.

What about if you have text? How does the word still become a number? Now that's a little bit harder. Take any word and map it to be a vector, e.g., the word "the" can be represented by the numbers [0 4 5 0 3 4]. We could assign arbitrary numbers to every word in the dictionary, but that is not ideal. It is not ideal because you would like to say that the word, the vector representation of plenty and the vector representation of many and the vector representation of much should all be relatively close to each other and should be very different from the vector representation of seeing because the word "seeing" and "plenty" are not as closely related as the word "much" and "plenty". This itself turns out to be a machine learning problem of assigning an appropriate vector to appropriate words in a language, but fortunately that's been done before. For a machine learning problem with freeform text as input, use word2vec, one of these techniques to convert words to vectors. Typically, there is a different word2vec for different languages, so you would go ahead and pick one of those word2vec objects and use it to convert your text into vectors. From there, you have numbers and use those as input to the machine learning problem.

Even though we are focusing primarily on structured data in this course, the things you do for images and text, the concepts still apply. As it turns out though, the kinds of layers and the kinds of models and the kinds of tricks that you would do for images and text are different, which is why we have this whole 10 course specialization on machine learning. In this course, we are going to stick to the basic concept of machine learning and applying it to structured data machine learning, which is the most common kind of data problem that you will face in most businesses. If you're going to learn machine learning, learn on structured data, because it tends to be the most useful. If you have a classification model, the labels would be spam or not spam, and the output of the model will not be 0 or 1, the output of the model will be a number between 0 and 1. It would be a floating-point number between 0 and 1. How do we interpret that number? What we do is because we decided the spam was going to be 1 and not spam was going to be 0, we'll interpret that number as a probability. That if this probability is 1 if it's 100 percent likely to be spam, and if the probability is 0, it is 100 percent likely to be not spam and if it's a number between 0 and 1, the number is 0.8, then we say that it is 80 percent likely that this image is spam. So, we interpret the output of the model, this output that's between 0 and 1, we interpret it as a probability of it being label equals 1.



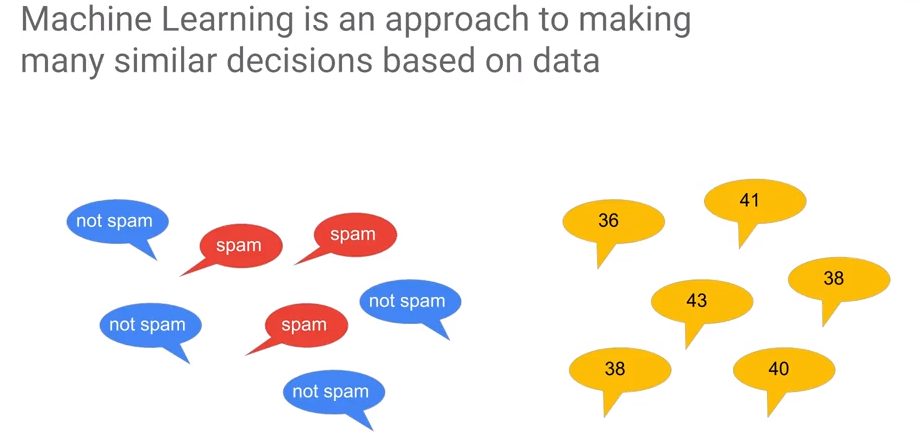
## Framing an ML Problem



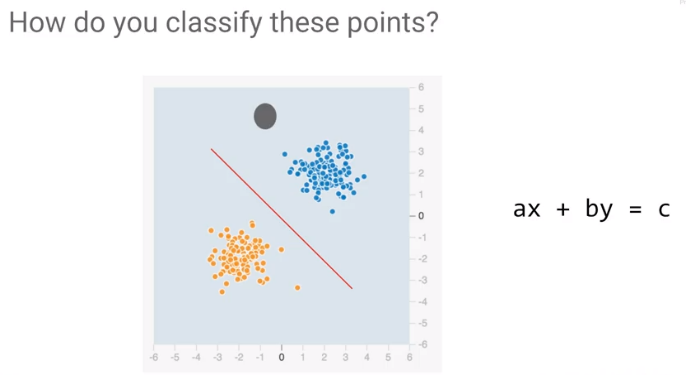
Consider this: you have a labeled data set, a data set of inputs, and the true known output for each of those inputs. Given such a training data set we can now create a machine learning model. The machine learning model is a mathematical function that will be able to create an approximation of that label for every one of those inputs. And then you can take that model and use it on inputs for which you are interested in knowing what the output ought to be. Given that framework, ML turns out to be extremely useful in a lot of different industries.

## Playing with Machine Learning

In this section, we develop an intuitive understanding of what neural networks are, what neurons are, what gradient descent is, etc.

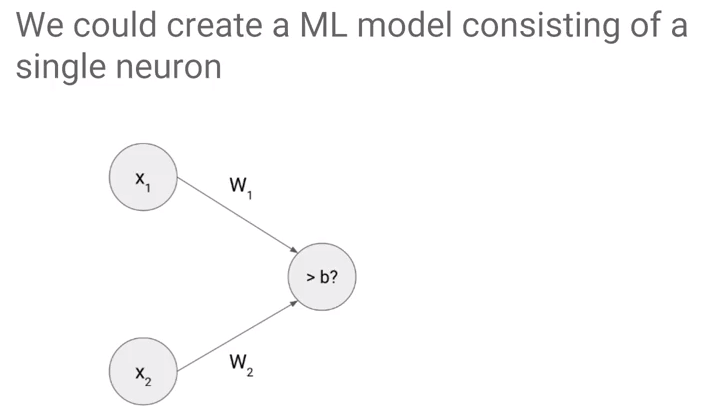
Machine learning is a way by which we make a lot of very similar decisions based on data. Machine learning is not about making one off decisions.

The statement above is generally correct. However, the author gave an example that is incorrect. He stated that the problem cannot be used to find delays in supply chain – this is the problem of finding anomalous behavior which can be handled well by classification. Indeed, many problems can be dealt with when using modern analytics.

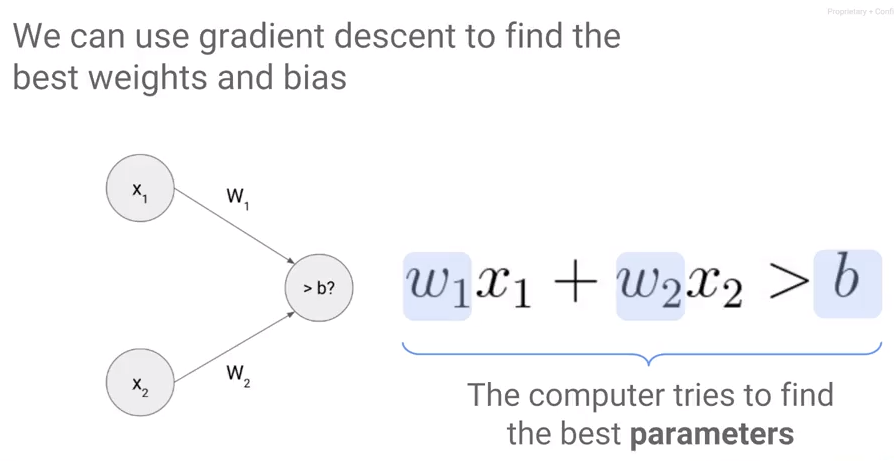
Machine learning is ideal for handling cases where outcome is consistent based on data.

For example, let the blue dots represent spam and the yellow dots are not spam. The topology of this problem makes it easy to find a solution. How would you classify these points? You would draw a line. And you would say that everything below this line is whatever the class is that corresponds to orange. And every point that's above this line is whatever the class is that corresponds to blue. Let's say all the blue are not spam. The equation of a line is ax + by = c you has two weights b and c. The intercept or bias is a constant. In terms of a neural network, if we have two inputs, x1 and x2, we can find a weight, w1, that we apply to x1, a weight, w2, that we apply to x2, and we can check if it's greater than some bias. That is, ax plus by, is it greater than c?

The object which collects the inputs and adds weights is called a neuron. A neuron is a way to combine all of the inputs and make a single decision on those inputs.

Graphically, a single neuron w1x1 + w2x2 is it greater than b, that single decision translates to be a line. This is the line separates the two sets of points.

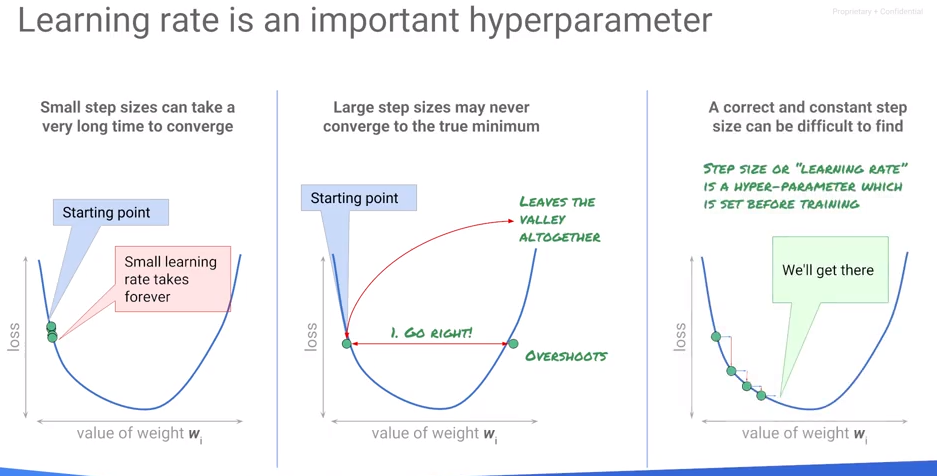
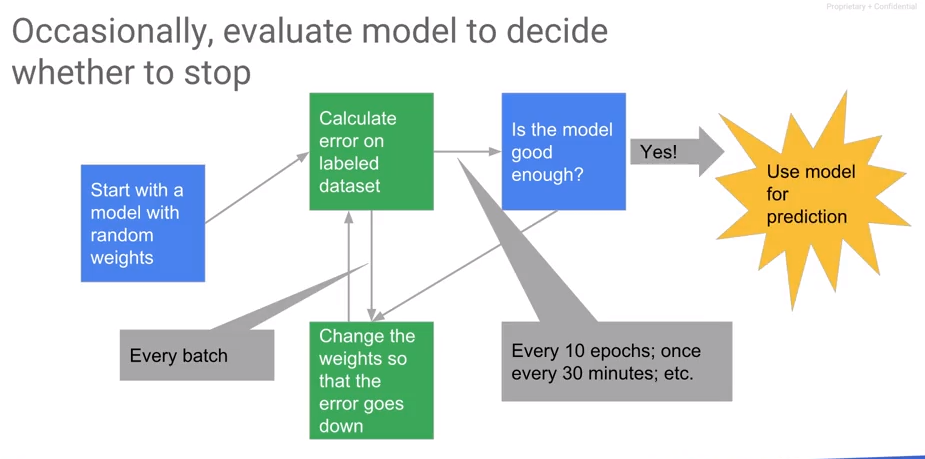
## Optimization



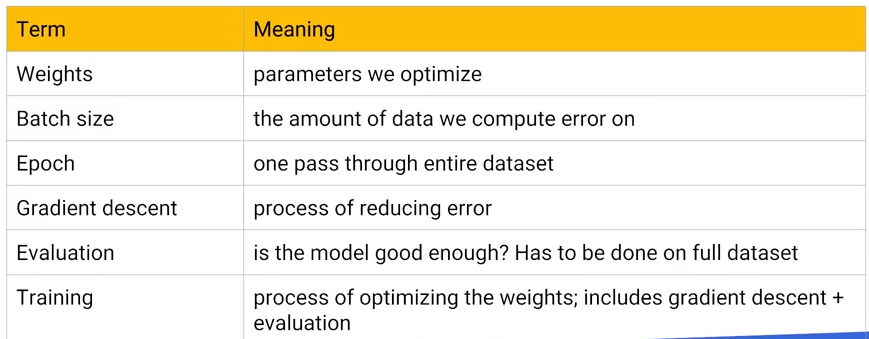
|  |
| --- |
| **Gradient Descent is Iterative** |

To find values for w1, w2 and b, use a process called gradient descent. With gradient descent, we walk down an error surface, hoping to end up at the place where the error is minimum. Visually, the idea being the more the points are distant from the separation line, the better it is.

Depending on what direction will reduce the error (moving us towards the minimum), we decide what the next value of the weight is going to be, because in this case, increasing the weight resulted in a lower error at the next iteration where we start is at that increased weight. So, we started the next iteration, we will repeat the process again and ultimately, we land up at the minimum. This isn't guaranteed though. Maybe when we change the weight, our change is so small that it takes a very large time to converge. The model takes a long time to train and so we give up and we don't actually end up at the best possible error. The size of the changes we make to the weights is called the learning rate. When the learning rate is low, training takes a very long time. If you try to speed things up by increasing the learning rate, we might miss the minimum altogether, so that's not good either. What we need is a perfect learning rate. The best learning rate unfortunately varies from problem to problem, so you will have to experiment.

Learning rate is an example of what is known as a hyper parameter. We look at hyper parameter tuning in a later module of this course. Let's look at the process of training a machine learning model. Start with random weights, calculate the error on a dataset for which you know the answers - the labeled dataset. Calculate the error on the label dataset and then we change those weights, and we try out different changes of weights in the neighborhood of that arbitrary weight that we started off with. Change them so that they're going to a direction where the arrow goes down, and then we go back and we calculate the error again. We tweak the weights and we calculate the error again, and we tweak the weights and we do this over and over again. Now when we do this tweaking of the weights, do we have to do it over the entire training dataset? Well, we could, but doing great in descent on the entire training dataset will tend to be quite slow. Your dataset may be many millions of rows long, and if every tweak requires you to go ahead and calculate the value of the error on all million rows, it will be quite slow. 

Instead, tweak the weights on a small batch of training data. Typical sizes of batches tend to be like 30 to maybe about 500 cycles. We calculate the errors, iterate through the weight changes, and then we have to decide when to stop. Because gradient descent is not guaranteed to converge, this gets to reasonable stopping point. Every once in a while, you want to determine if the model has converged sufficiently. Convergence should be checked every n epochs - an epoch is a traversal through the entire training dataset. If the model is good enough, then you stop, you export the model and you use the model for prediction.



## A Neural Network Playground – Choosing Network Configuration

[http//playground.tensorflow.org](http/playground.tensorflow.org)

I threw out the complete lecture from Lack because he is just repeating himself again and again. The bigger question which he is likely not going to get to is how to choose the topology of the network. I started playing with the spiral dataset. My intuition was that the more layers and neurons I throw at the problem, the better. But it was not so – I found that one hidden layer with five neurons worked about as well as anything else.

What kind? Artificial Neural Network (ANN)? Deep Neural Network (DNN)? Convolutional Neural Network (CNN)? Recurrent Neural Network (RNN)? And within RNNs, Long Short-Term Network (LSTM) or Gated Recurrent Units (GRU)? What about Generative Adversarial Networks (GAN)?

To simplify - how do you choose the topology of a (potentially deep) neural network (ANN or DNN)?

See <https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw> and <http://dstath.users.uth.gr/papers/IJRS2009_Stathakis.pdf>. See also <https://stats.stackexchange.com/questions/222883/why-are-neural-networks-becoming-deeper-but-not-wider>.

#### Summary of Points