Google Cloud

Getting started with Machine Learning

Data Engineering on Google Cloud Platform

Notes:

2 hours lecture + 30 min of interactive quizzes interspersed + 30 min lab. Be smart about the timeline for this section. It will take the full 3 hours for an audience that is new to ML but should take less time if they already know some of this. Before you start diving into the slides, ask questions like:

(1) Do you know what ML is?
(2) Have you built a ML model before? What toolkit did you use?
(3) Do you know what deep neural networks are?

Based on the answers to these questions, pace this material appropriately. For example, you should plan on covering this chapter in 60 minutes if the audience has prior experience with ML -- don't belabor the point and feel free to skip slides.

Be smart about how deep to cover this section:

(1) Normal pace
(2) Cover only slides 6 and 17, using them to summarize the main points of section.
(3) Skip this section completely.

Choose option #1 if these are developers new to ML. Choose #2 if these are enthusiasts who have read about ML already but have not built a ML model for their own use cases. Choose #3 if these are people who have used scikit-learn etc.

Machine Learning is a way to use standard algorithms to derive predictive insights from data and make repeated decisions

data

algorithm

predictive insight

decision

ML is a way to derive *predictive* insights from data.
We do this using algorithms that are relatively general and applicable to a wide variety of datasets.
Think back to your company. How do you use data today? Perhaps you have a dashboard that business analysts and decision makers view on a daily basis? Perhaps a report that they read on a monthly basis? That is an example of backward-looking use of data -- looking at historical data to create reports and dashboards; that tends to be what people mean when they talk about business intelligence. A lot of data analytics is backwards-looking.

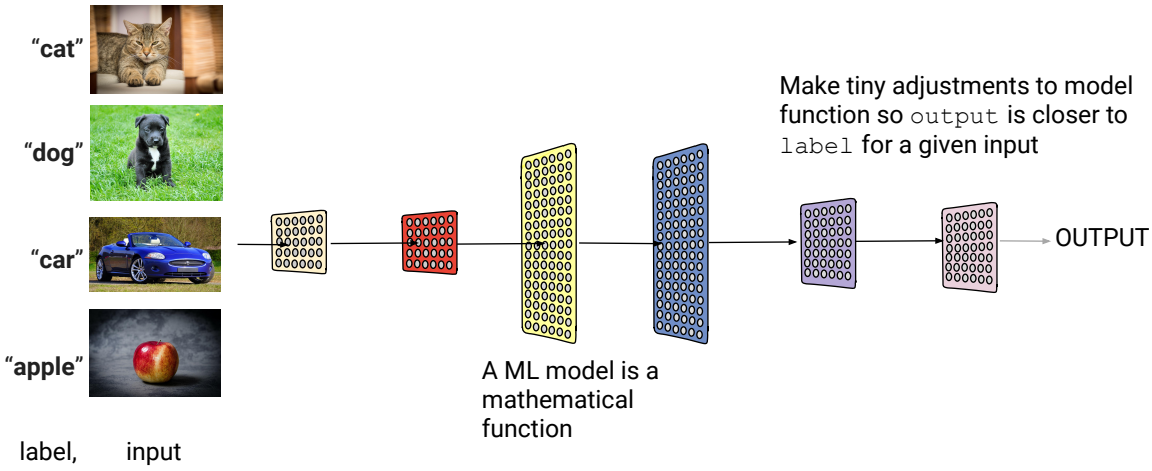
Of course, the point of looking at historical data might to be make decisions. Perhaps business analysts examine the data and suggest new policies or rules? They suggest, for example, that it might be possible to raise the price of a product in a certain region. Now, the business analyst is making a predictive insight. But is that scalable? Can the business analyst make such a decision for every product in every region? Can they dynamically adjust the price every second? In order to make decisions around predictive insights repeatable, you need machine learning. You need a computer program deriving such insights.

So, machine learning is about making many predictive decisions from data. It is about scaling up business intelligence and decision making.

https://pixabay.com/en/hard-disk-storage-computer-159264/ (cc0)

https://pixabay.com/en/fractal-pattern-abstract-form-142748/ (cc0)
https://pixabay.com/en/arrows-inside-pressure-request-2029157/ (cc0)
https://pixabay.com/en/sign-direction-kids-cute-paint-2792576/ (cc0)

Stage 1: Train an ML model with examples



Notes

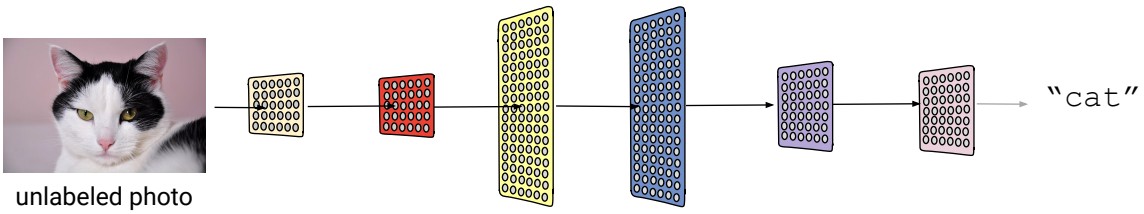
The first stage of ML is to train an ML model with examples. The form of machine learning that we will be focused on in this specialization is called supervised learning, and in supervised learning, we start from examples.
An example consists of a label and an input. For example, suppose we want to train a machine learning model to look at images and identify what is in those images. The true answer is called the label, so "cat" for the first image and "dog" for the second image. The image itself, the pixels of the image, are the input to the model.
The model itself is a mathematical function, of a form that can be applied to a wide variety of problems.
The models used in ML have a bunch of adjustable parameters. Then when we "train" the model, what we do is to make tiny adjustments to the model so that the output of the model -- the output of the mathematical function -- is as close as possible to the true answer for a given input. Of course, we don't do this on one image at a time -- the idea is adjust the mathematical function so that overall, the outputs of the model for the set of training inputs is as close as possible to the training labels.

The key thing is that machine learning, at least ML of the form that we will consider in this course (the most mature form of ML), relies on having a dataset of labeled examples. Labeled examples -- input + the true answer.

https://pixabay.com/en/domestic-cat-cat-adidas-relaxed-726989/ (cc0)

https://pixabay.com/en/dog-young-dog-puppy-280332/ (cc0)
https://pixabay.com/en/sports-car-vehicle-transportation-1317645/ (cc0)
https://pixabay.com/en/apple-education-school-knowledge-256268/ (cc0)

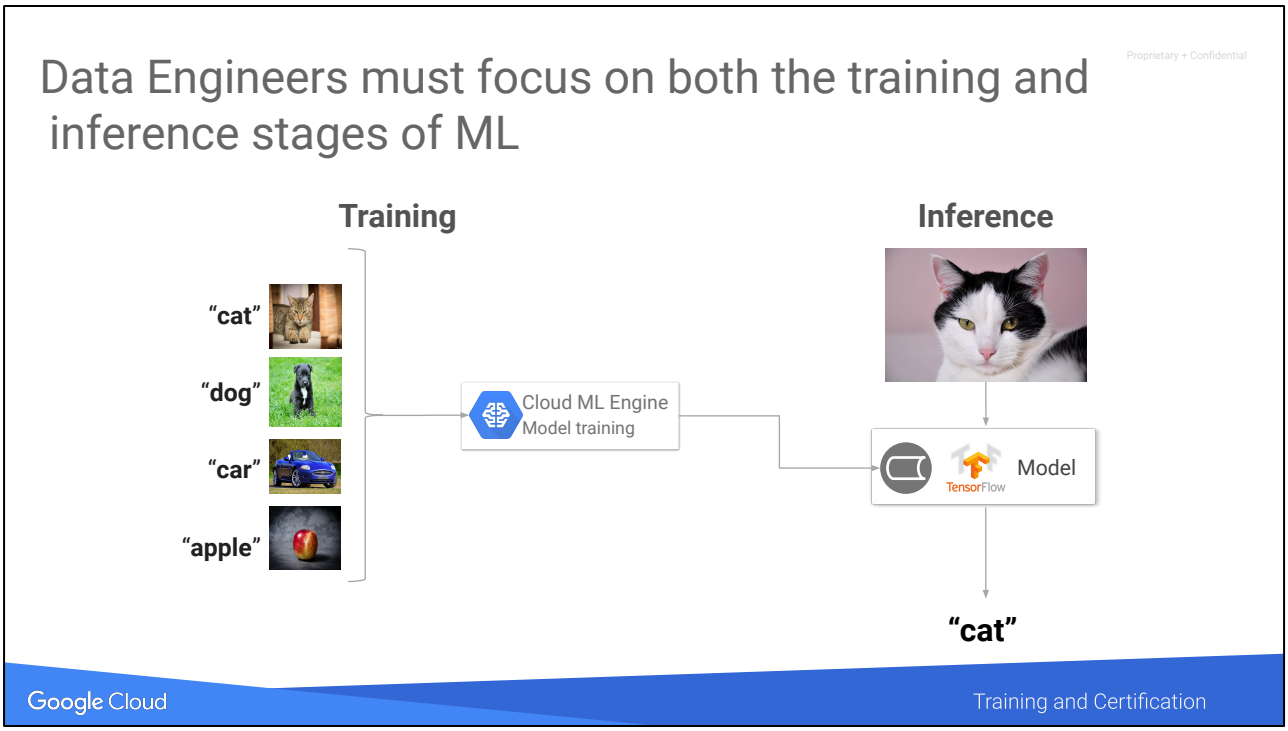
Stage 2: Predict with a trained model



Notes
And after the model is trained, we can use it to “predict” the label of images that it has never seen before.
Here, we are inputting to the trained model this image and because the network has been trained, it is correctly able to output “cat”.
Note that the cat image on this slide is different from the one before it -- still works because the ML model has “generalized” from the specific examples of cat images we showed it to a more general idea of what a cat is and what it looks like.

The key to making a ML model generalize is data and lots and lots of it.
Having labeled data is a precondition for successful ML.

https://pixabay.com/en/cats-a-normal-cat-pet-796437/ (cc0)



It is important to realize that machine learning has two stages: training and inference. Sometimes, people refer to "prediction" as "inference" because "prediction" seems to imply a future state and in the case of images like this, we are not really "predicting" that it is a cat, just "inferring" that it is a cat based on the pixel data.

It can be tempting, as a Data Engineer, to focus all your energy, on the first stage. But this is not enough.
You need to be able to operationalize the model, put the model into production so that you can run inferences.

Look at many books on machine learning, blog posts, university courses ... They tend to ignore this second stage of ML. But in the real-world, what is the use of training a ML model if you can not use it?
In this specialization, we will be careful to show you machine learning, end-to-end. And by end-to-end, we mean putting ML models into production.

https://pixabay.com/en/cat-cute-cat-baby-kitten-pet-1992140/(cc0)
https://pixabay.com/en/mouse-rodent-cute-mammal-nager-1708347/

Do Now: In your own words, write down definitions for these ML terms

Term	Meaning
Label	
Input	
Example	
Model	
Training	
Prediction	

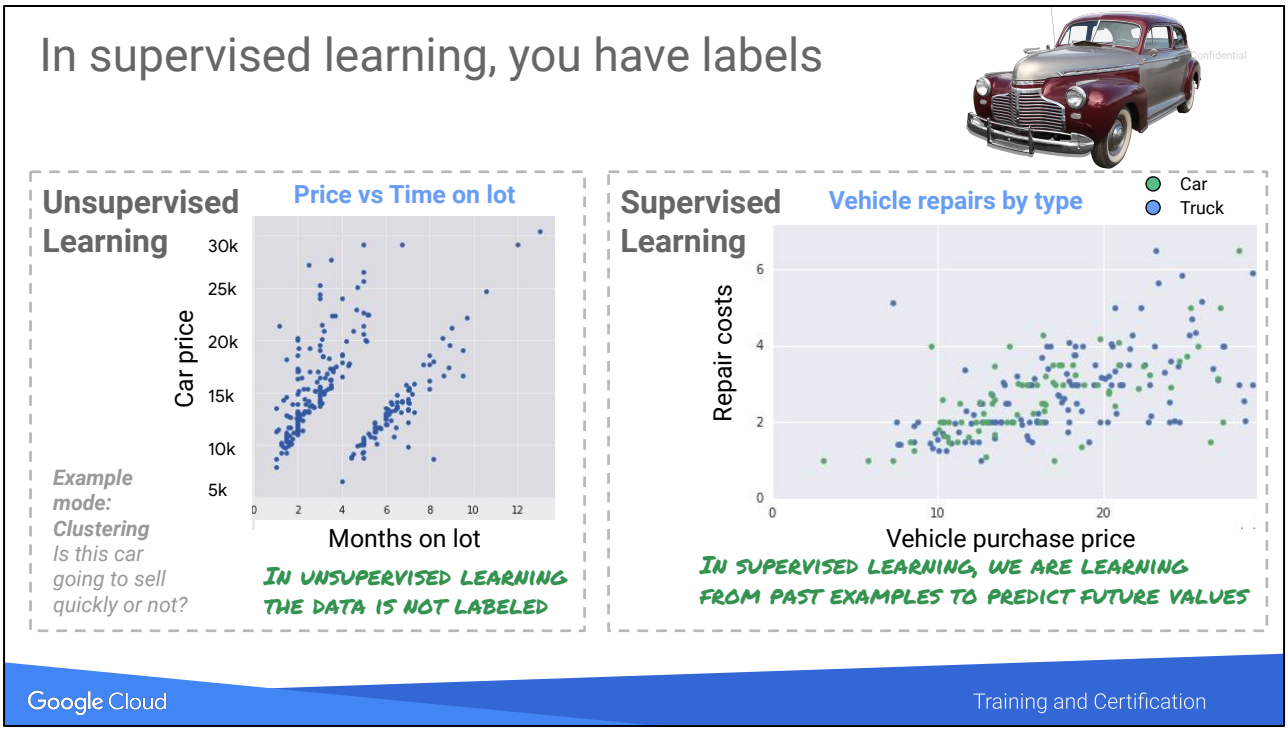
Notes:

Label = true answer
Input = predictor variable(s), what you can use to predict the label
Example = input + corresponding label
Model = math function that takes input variables and creates approximation to label
Training = adjusting model to minimize error
Prediction = using model on unlabeled data

Students are likely to struggle with the idea of a model. So, in the recap to this slide, delve more into models. So explain a ML model as a math function that weights & adds inputs and does mathematical transformations like tanh() to the numbers. And tell them that we will look at it again and again.

- That ML is a function can be hard for attendees to see. As an example, assume that the first layer returns the number of pixels that are brown/black/blue/red, and the second layer finds the most common color, and the third layer returns “cat” if previous layer had supplied “brown”. Based on just these images, that is a perfectly valid ML model. With more data, we’ll have to do more finely tuned ML functions.
- Now, if you know ml you are probably skeptical that my “function” above is really achievable with a neural network, so consider this.

- Mathematically, this model would be, for the first layer, [sum(r = 255, g=255, b=255), …, …, sum(r=255, g=0, b=0)] -- this is just a set of appropriately positioned relu functions (okay, for r=234, we'd need two relu functions, so two layers, but you get the idea). The second layer would be a softmax layer. The third layer is simply an identity! [Instructors: don't go into this since it will only confuse a class … if challenged, ask the person doing the challenging to read the slide notes.]



Notes

Two of the most common classes of machine learning models are supervised and unsupervised ML models. The key difference is that with supervised models, we have labels, or in other words, the correct answers to whatever it is that we want to learn to predict.

In unsupervised learning, the data does not have labels.

The graph on the left is an example of the sort of problem that an unsupervised model might try to solve. You own a used car lot. Heye, you want to look at the months that a car remains on the lot and its price, and then group or cluster cars, to see whether a car is likely to sell quickly. Critically, there is no ground truth here; the car lot does not have one group of cars that it is trying to sell and another group of cars it plans to hold. Consequently, unsupervised problems are all about discovery, about looking at the raw data, and seeing if it naturally falls into groups. At first look, it seems that there are two distinct clusters or groups that could be separated nicely with a line.

In this course though, we'll be focused on supervised machine learning problems, like this one. The critical difference is that with supervised learning, we have some notion of a "label," or one characteristic of each data point that we care about a lot.

Typically, this is something we know about in historical data, but we don't know in real time. We know other things, which we call predictors, and we want to use those

predictors to predict the thing we don’t know.

For example, let's say you are a purchaser for the used car lot. You have historical data on vehicle purchase prices for cars and trucks and how much was spent on repairs before the vehicle could be sold. Now you are at an auction. And you are looking at a vehicle that is for sale. You know the asking price and you know whether it is a car or a truck. But you don't know how much the repairs are going to be. In the historical data, the repair expense is a label. You create a model to predict the repair expense from the asking price. Then you try to predict the repairs, in real time, based on historical data and the values you know for a specific vehicle. This will give you an indicator of whether the asking price is a good deal for the vehicle or if they are asking too much because of the repair expenses. You will have an indicator how to bid at the auction.

https://pixabay.com/illustrations/chevrolet-limousine-usa-pkw-2178828/

Regression and classification are supervised ML model types

Price	Repairs	Type	Accident	Year	Odometer
43k	1,012.14	Truck	No	2012	53,232
6k	830.12	Car	No	2003	140,091
18k	2,704.87	Car	Yes	2017	6,508
3k	3,153.04	Truck	No	1999	162,000
35k	205.33	Car	No	2015	23,122

Model 1
Regression

Model 2
Classification

Google Cloud Training and Certification

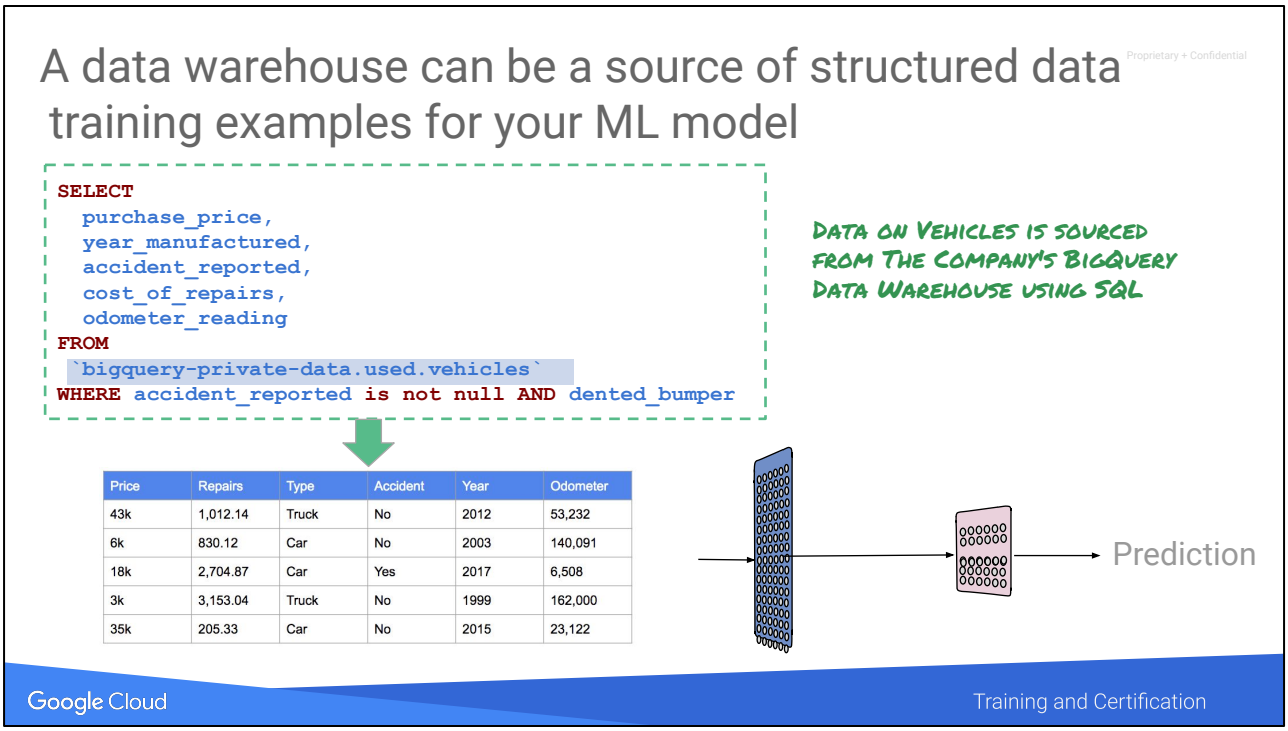
Within supervised ML there are two types of problems: regression and classification. To explain them, let's dive a little deeper into this data.

In this dataset of repair expenses, each row has many characteristics, such as type of vehicle, whether it has been involved in an accident, the year it was manufactured, and the odometer reading indicating how far it has traveled. We'll choose one of the columns as the characteristic we want to predict, called the "label". And we choose a set of the other columns which are called the "features".

In new model 1, we want to predict the repair amount. Therefore, the column "Repairs" is my label. I can use one, all, or any number of the other columns as my features to predict the repairs.
This will be a regression model because repairs is a continuous label.

In new model 2, we want to predict the whether the vehicle has been in an accident. Once again, I will use some set of the rest of the columns as features and try to predict the accident field. This will be a classification model because the label has a discrete number of values or classes.

https://pixabay.com/photos/pontiac-usa-wreck-dare-corrosion-2257344/



Notes

Now, where does this data come from? The dataset is what we call structured data -- consisting of rows and columns -- and a very common source of structured data for machine learning is your data warehouse. Unstructured data is things like pictures, audio, or video.

Here I'm showing you a vehicle dataset, a private dataset of company information. It is a dataset in BigQuery that is in your data warehouse

Let's say we want to predict the Odometer reading of the vehicle. In other words, we want to predict how far the vehicle has been driven.

You can do a SQL SELECT statement in BigQuery to create a ML dataset -- we will choose input features to the model, things like purchase price, accidents reported, manufacturing year, and type of vehicle.

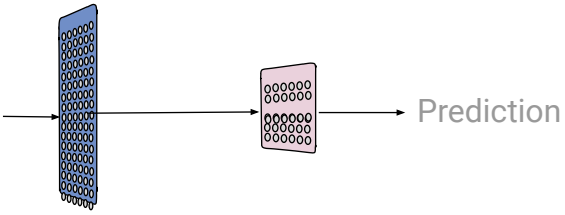
Because the Odometer reading is a continuous value, this is a regression problem.

Making predictions from structured data is very commonplace, and that is what we focus on in the first part of the specialization.

Since repairs is a continuous value, use regression to predict

Price	Repairs	Type	Accident	Year	Odometer
43k	1,012.14	Truck	No	2012	53,232
6k	830.12	Car	No	2003	140,091
18k	2,704.87	Car	Yes	2017	6,508
3k	3,153.04	Truck	No	1999	162,000
35k	205.33	Car	No	2015	23,122

COST OF REPAIRS IS STORED AS A FLOATING POINT NUMBER, REPRESENTING A CONTINUOUS (REAL) VALUE

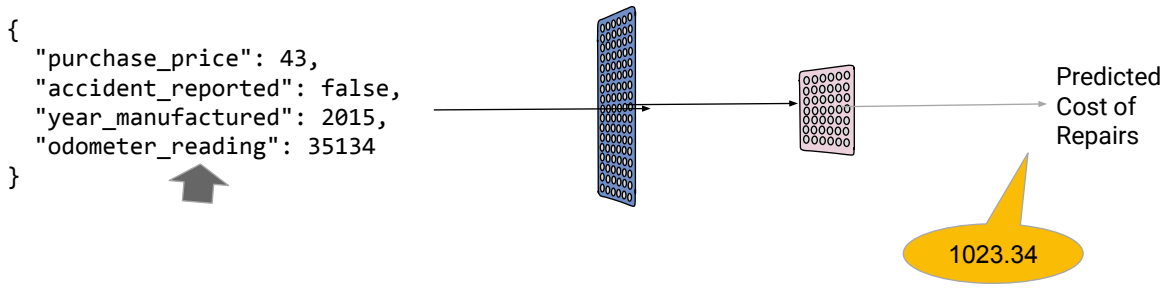


REGRESSION DNN MODEL

This dataset can be used to predict the cost of repairs.

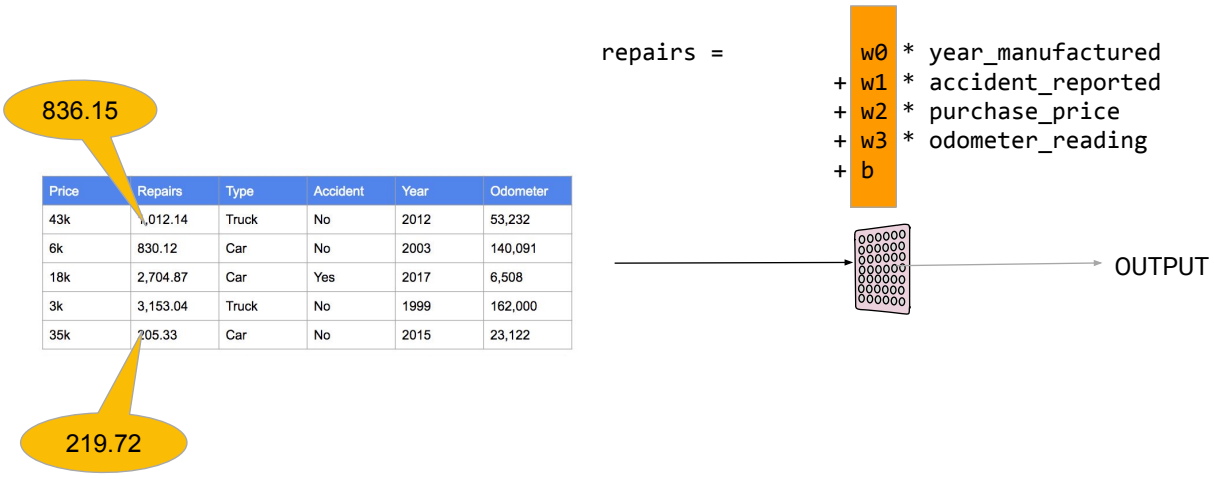
Perhaps we want to predict cost of repairs. And we want to predict this using other attributes of the data as features.
The label here would be Repairs, and it is a continuous variable. It is stored as a floating point number, which would make this a regression problem.

The model is fed information collected in real-time, and used for prediction



The key point here is that you are predicting from real-time data, for example, from Cloud Pub/Sub or your web application.
This information might originate from an online car auction service.

The model may even have only one layer



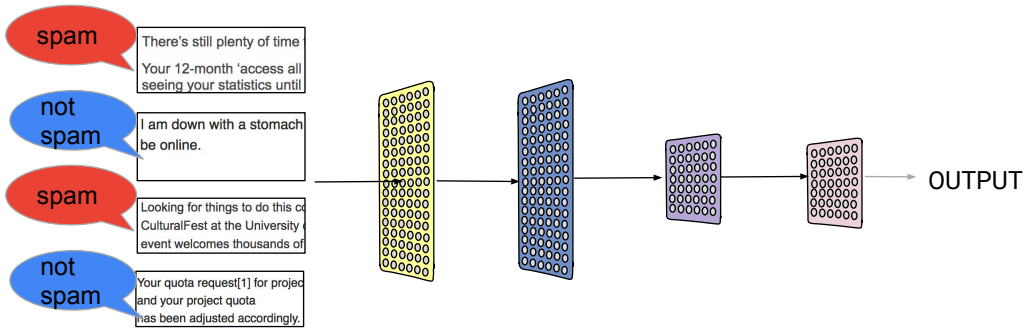
Notes:

Training involves finding weights w0, w2, w2, w3 such that the predicted rate (fuel_efficiency) is really, really close to the labels.

A neural network with no hidden layers is a linear model …

- Point out that the model here is a lot simpler than the one on the previous slides -- more layers in a DNN makes it more complex. Models on structured data are typically only a few layers deep whereas image classification DNNs can have hundreds of layers.

A classification model can be used to detect whether email is spam or not



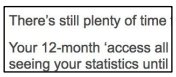
The input here is text

Notes:

- Key words on this slide: “classification” -- use these words in a sentence.
- Say that is another classification problem, similar to the dog/cat/ example. In this case, we have only two categories or labels.
- The input here does not seem numeric ...

The inputs for unstructured data are still ultimately just numbers

N-dimensional array of pixel values



Each word is mapped to a vector
e.g., "the" could be [0 4 5 0 3 4]
Coming up with an appropriate vector for a word is itself a machine learning problem

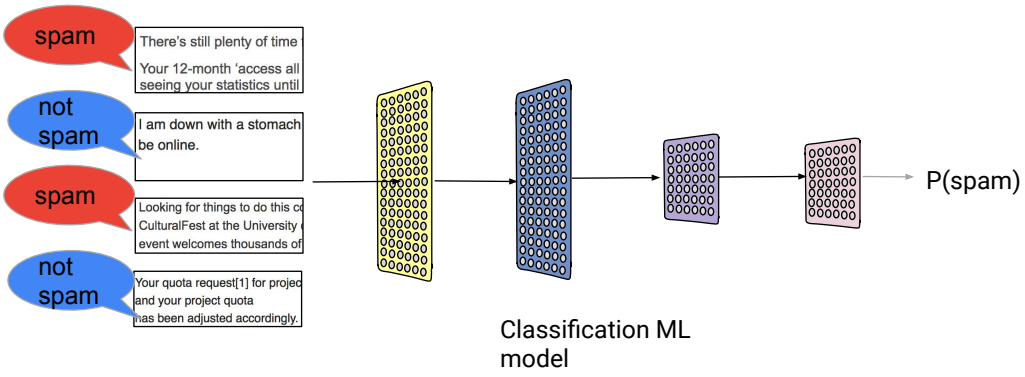
Notes:

Pixel values here are [r, g, b, a] and the image itself is 2D, so it could be a width x height x 4 array.

word2vec

https://pixabay.com/en/rocket-launch-smoke-rocket-take-off-67723/ (cc0)

The output of the model might be the probability that the email is spam



Notes:

- Note the probabilistic output -- we would have to threshold the probability if we wanted “spam” or “not spam”. The previous model returned a probability for each label (cat, dog, etc.) and we picked the most likely (usually, we also threshold; i.e., we don’t return “cat” if the P(cat) is only 0.01).

Machine Learning used in lots of industries

Manufacturing

- Predictive maintenance or condition monitoring
- Warranty reserve estimation
- Propensity to buy
- Demand forecasting
- Process optimization
- Telematics

Retail

- Predictive inventory planning
- Recommendation engines
- Upsell and cross-channel marketing
- Market segmentation and targeting
- Customer ROI and lifetime value

Healthcare and Life Sciences

- Alerts and diagnostics from real-time patient data
- Disease identification and risk satisfaction
- Patient triage optimization
- Proactive health management
- Healthcare provider sentiment analysis

Travel and Hospitality

- Aircraft scheduling
- Dynamic pricing
- Social media—consumer feedback and interaction analysis
- Customer complaint resolution
- Traffic patterns and congestion management

Financial Services

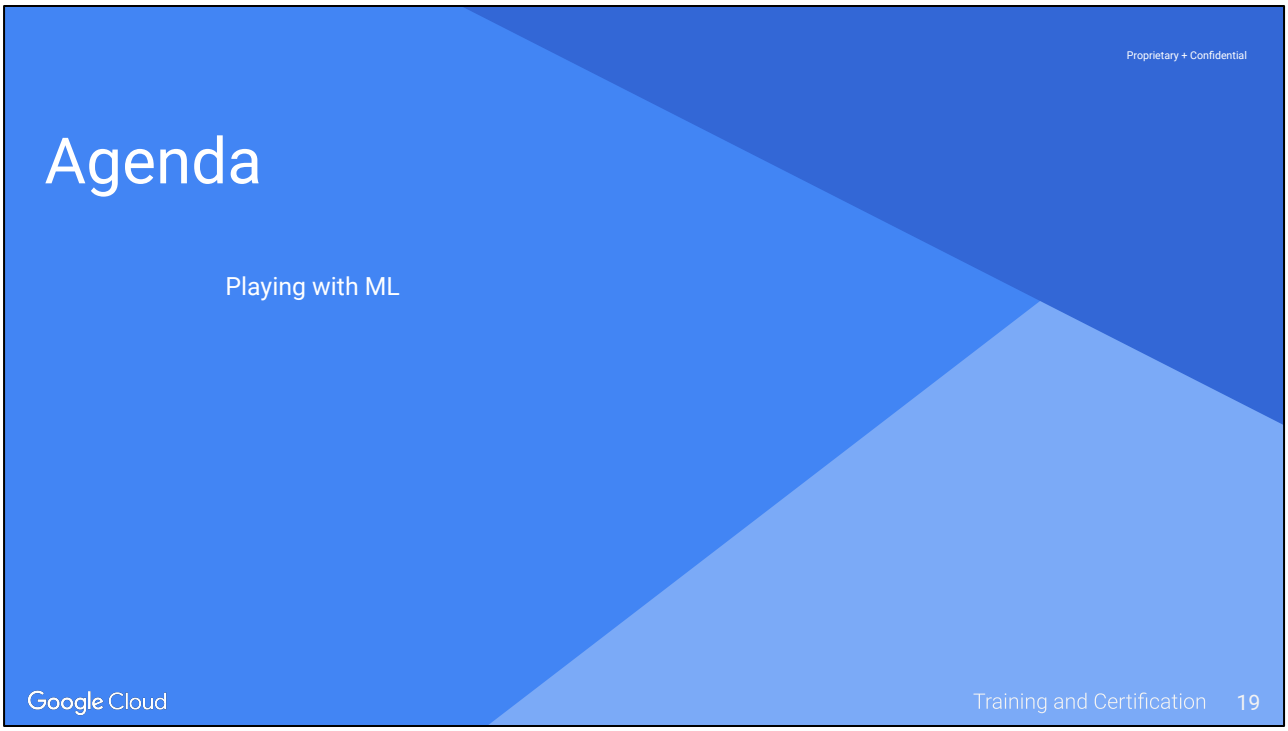
- Risk analytics and regulation
- Customer Segmentation
- Cross-selling and upselling
- Sales and marketing campaign management
- Credit worthiness evaluation

Energy, Feedstock and Utilities

- Power usage analytics
- Seismic data processing
- Carbon emissions and trading
- Customer-specific pricing
- Smart grid management
- Energy demand and supply optimization

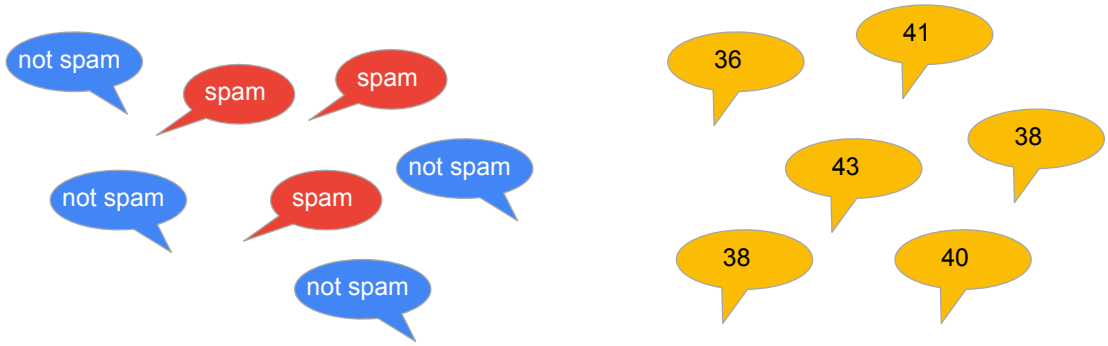
Do Now: Pick 5 use cases from previous slide and fill out this table

Use case	Label	Input(s)	Classification or regression?



This section should take 20-40 minutes based on audience. In my experience, even people who know ML get something out of this section because it introduces neural networks, deep networks, and feature engineering in an intuitive way. However, the more sophisticated the audience, the faster you should cover this.

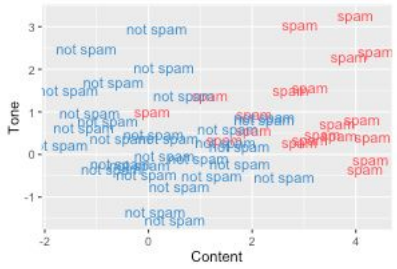
Machine Learning is an approach to making many similar decisions based on data

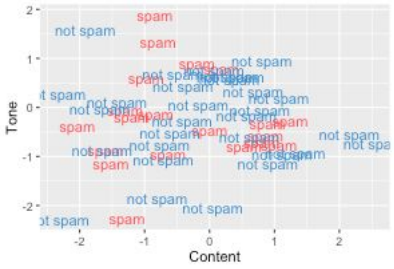


Notes:

You train the computer using lots of examples. On the left is spam-filtering. On the right, the gestation model.
This slide is here to make more obvious the relationship between the “real” examples in previous section and the “toy” example in this one.

ML = Pattern recognition from examples



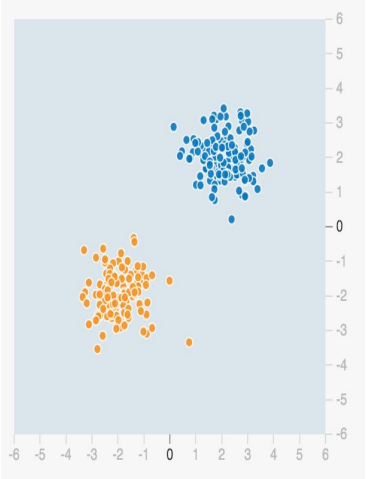


Notes:

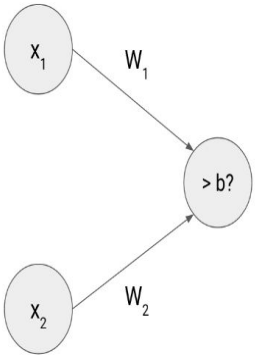
The right-hand side is a more difficult problem than the left-hand side, but with non-linear models and transformations, the ML model can usually manage to find a way to separate the two classes to varying levels of accuracy. Use the right-hand side to lead into a discussion on accuracy, which leads into next slide. Also, tie in the blue-red here to the dots on the next slide.

Graphs created by Cassie Kozyrkov, Google

How do you classify these points?



We could create a ML model consisting of a single neuron



$$w_1 x_1 + w_2 x_2 > b$$

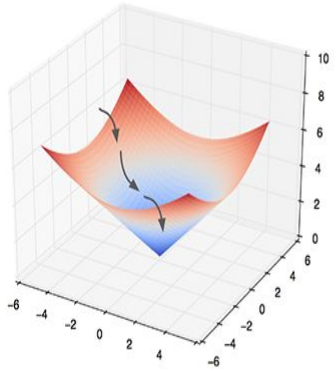
Graphically, that translates to: find a line that separates the two sets of points

bias
(threshold)

$$w_1 x_1 + w_2 x_2 > b$$

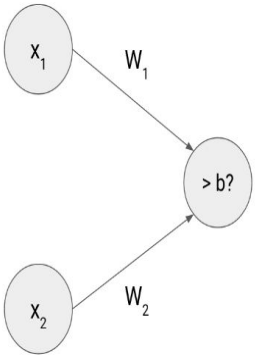
weights

Gradient descent is used to find the best parameters



	-5	-4	-3	-2	-1	0	1	2	3	4	5
5	50	41	34	29	26	25	26	29	34	41	50
4	41	32	25	20	17	16	17	20	25	32	41
3	34	25	18	13	10	9	10	13	18	25	34
2	29	20	13	8	5	4	5	8	13	20	29
1	26	17	10	5	2	1	2	5	10	17	26
0	25	16	9		1	0	1	4	9	16	25
-1	26	17	10	5	2	1	2	5	10	17	26
-2	29	20	13	8	5	4	5	8	13	20	29
-3	34	25	18	13	10	9	10	13	18	25	34
-4	41	32	25	20	17	16	17	20	25	32	41
-5	50	41	34	29	26	25	26	29	34	41	50

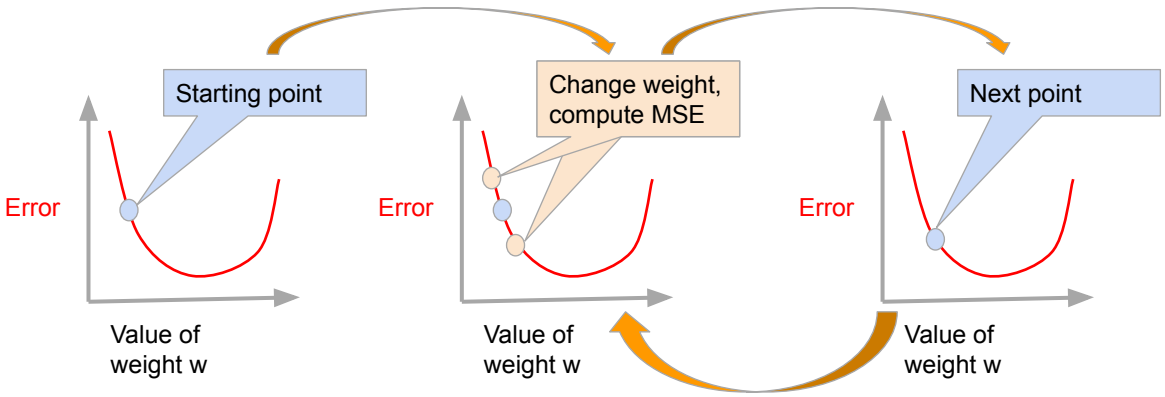
We can use gradient descent to find the best weights and bias



$$w_1 x_1 + w_2 x_2 > b$$

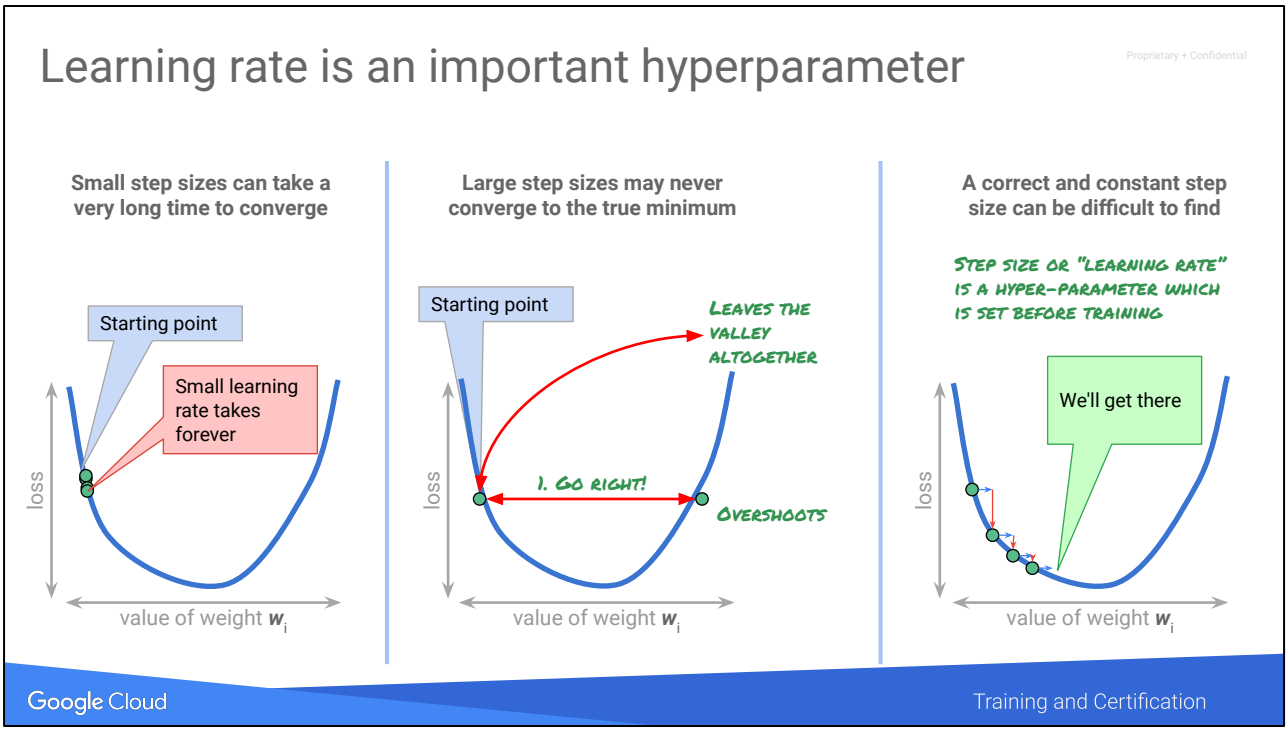
The computer tries to find the best **parameters**

Recompute error after each batch of examples (not full dataset)



Notes:

Typical batch size = 100-500 samples.

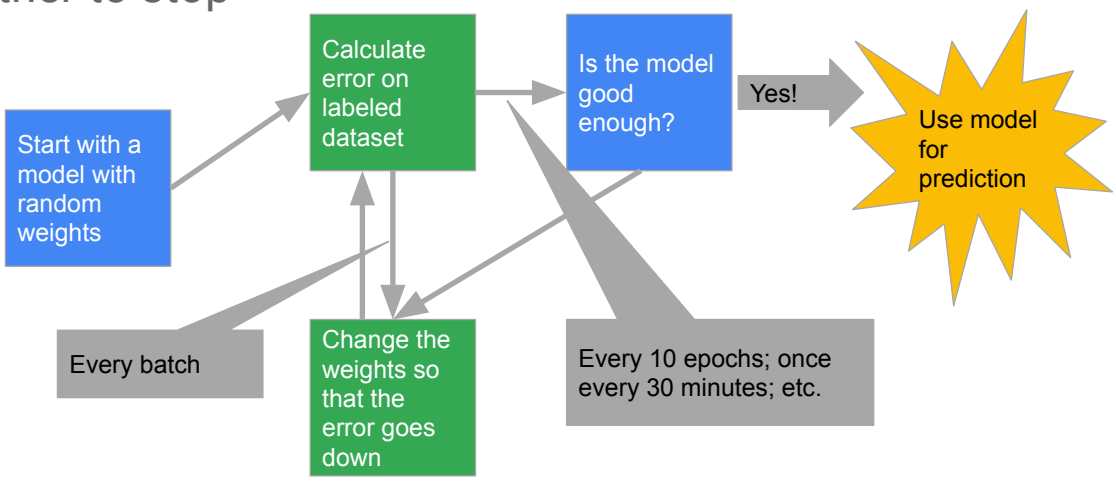


Small step sizes: Putting aside direction for the moment, if your step size is too small, your training might take forever. You are guaranteed to find the minimum though (so long as there is only one minimum like this linear regression loss curve here -- remember our two minimum loss surface we just showed previously? We'll cover how to deal with those efficiently later).

Larger step sizes: If your step size is too big, you might either bounce from wall to wall or bounce out of your valley entirely, and into an entirely new part of the losos surface. Because of this, when the step size is too big, the process is not guaranteed to converge.

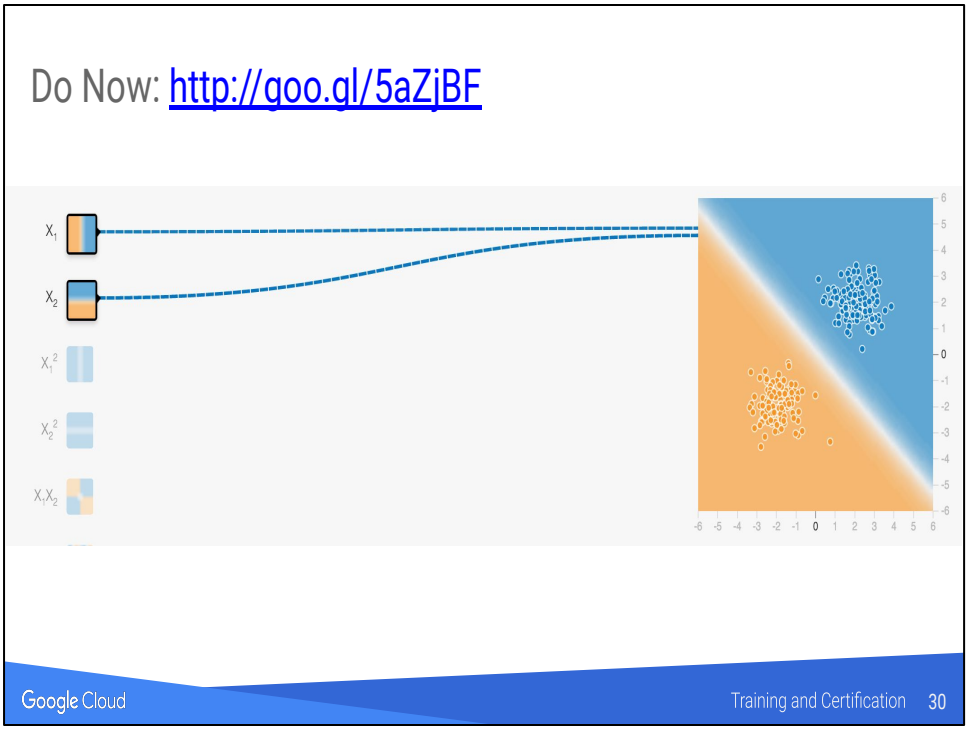
Correct step size: If your step size is just right …. Well, then, you're set. But whatever this value is, it's unlikely to be just as good on a different problem.

Occasionally, evaluate model to decide whether to stop



Notes:

Note that after we calculate error on batch, we can either keep going or we can evaluate the model.
Evaluating the model needs to happen on full dataset, not just a small batch!



Notes:

Playground URL for this pattern: http://goo.gl/5aZjBF

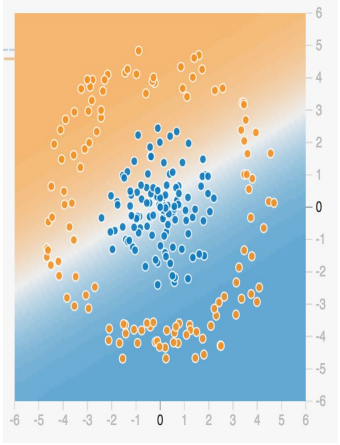
Do Now: In your own words, write down definitions for these ML terms

Term	Meaning
Weights	
Batch size	
Epoch	
Gradient descent	
Evaluation	
Training	

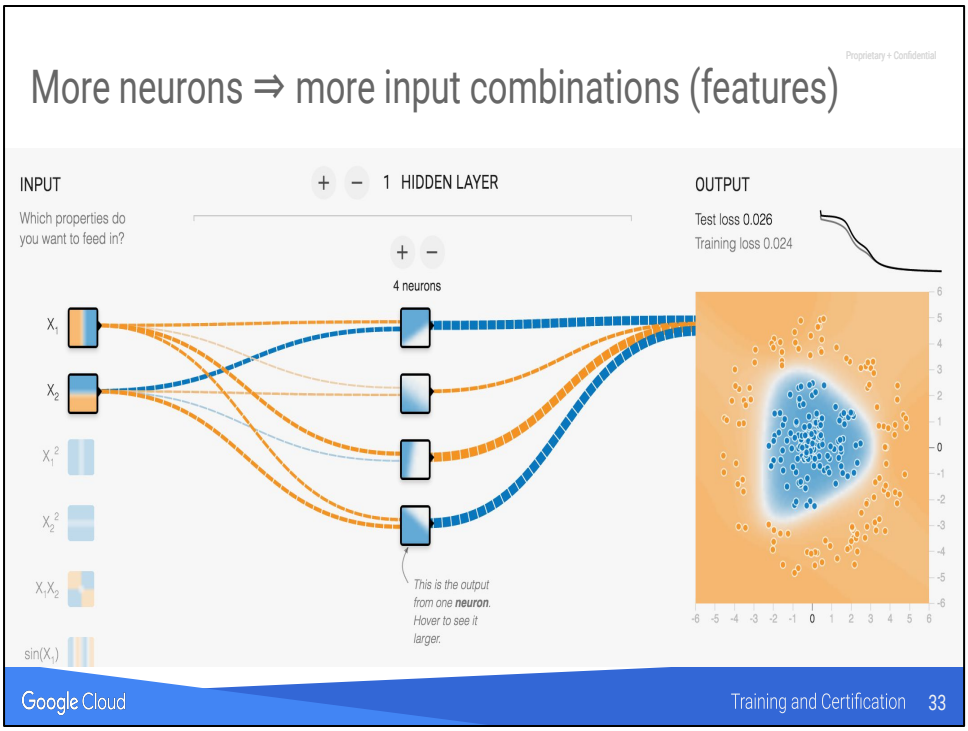
Notes:

Weights/bias = parameters we optimize
Batch size = the amount of data we compute error on
Epoch = one pass through entire dataset
Gradient descent = process of reducing error
Evaluation = is the model good enough? Has to be done on full dataset
Training = process of optimizing the weights; includes gradient descent + evaluation

Do Now: Can you use a single line to separate these? What do you have to do?



http://goo.gl/v7qM4Q

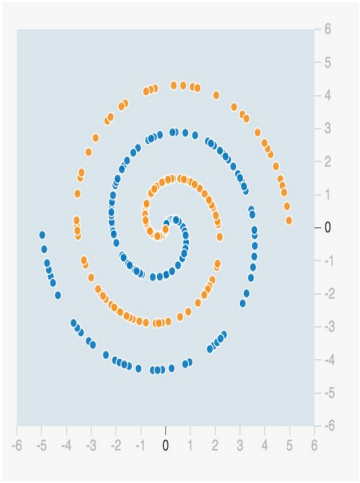


Notes:

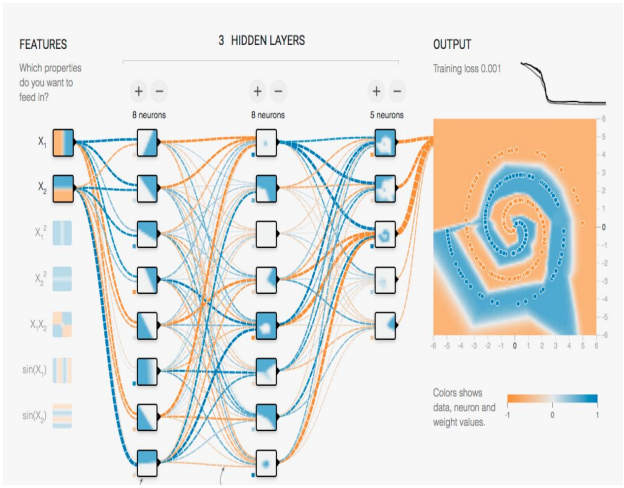
Note that the separation boundary is a polygon … each side is a linear combination of features …. We should have gotten a 4-sided polygon, but in the diagram above two of the sides merged (i.e., their weights were linearly correlated).

Playground URL for this pattern: http://goo.gl/rSX7Ve

How about this? Will a set of lines work?



More hidden layers ⇒ more hierarchies of features

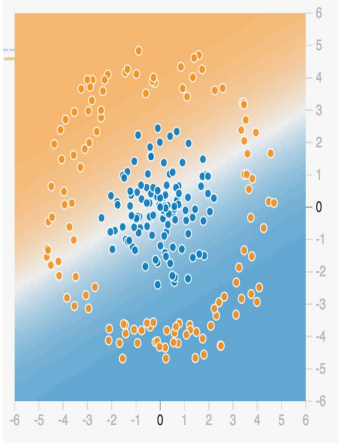


http://goo.gl/fymPMI

Notes:

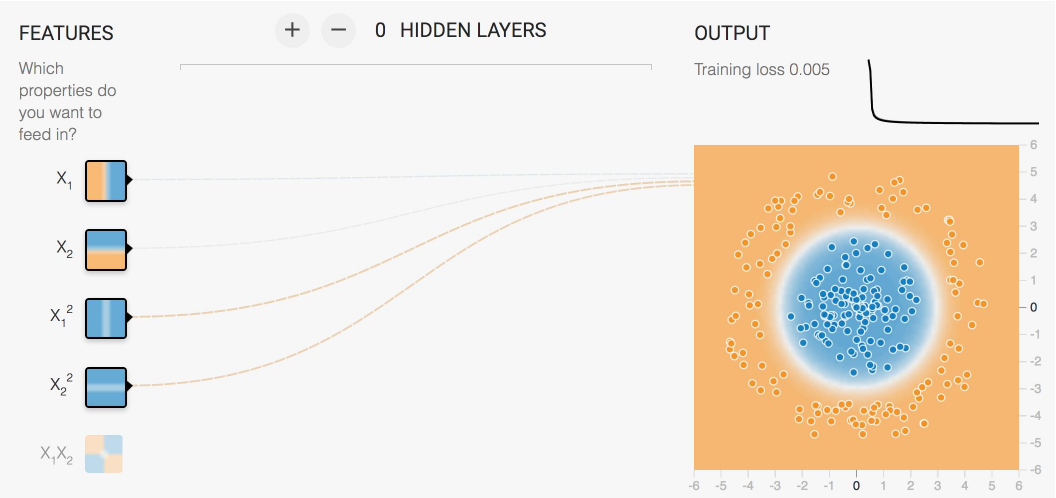
Even a complex non-linear pattern such as the double spiral could be classified by the simple neural network.

Do Now: Can you use a single line to separate these without adding layers?



http://goo.gl/v7qM4Q

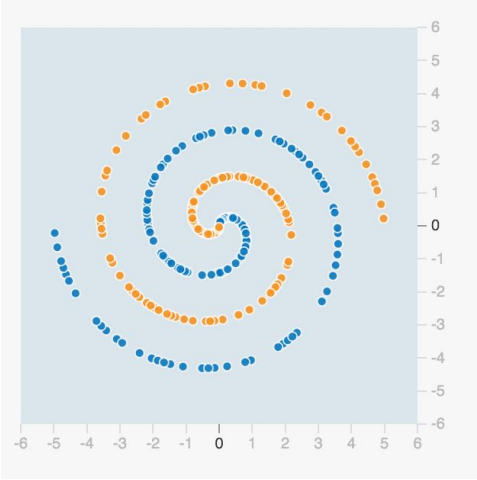
Engineer some extra features



Notes:

The reason we know that x^2 and y^2 will work is because we realize that points close to the origin are blue and those far away are orange. This indicates that distance would be a good feature, hence x^2 + y^2 ...

These features help in the spiral case too...



http://goo.gl/jdvKga

Do Now: In your own words, write down definitions for these ML terms

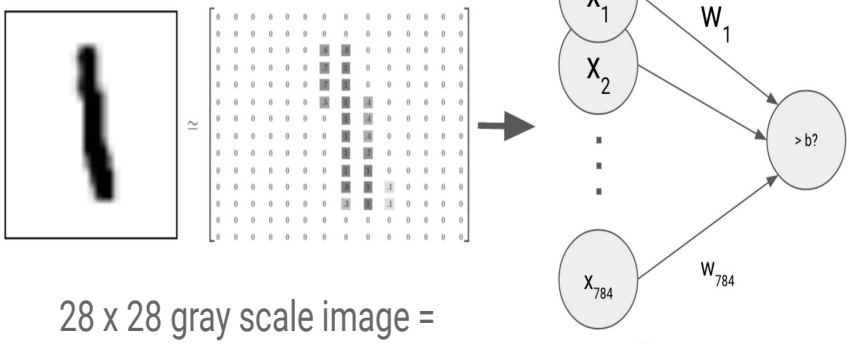
Term	Meaning
Neurons	
Hidden layer	
Inputs	
Features	
Feature Engineering	

Notes:

Neuron = one unit of combining inputs
Hidden layer = set of neurons that operate on the same set of inputs
Features = transformations of inputs, such as x^2
Feature engineering = coming up with what transformations to include

What about images? How does it work?

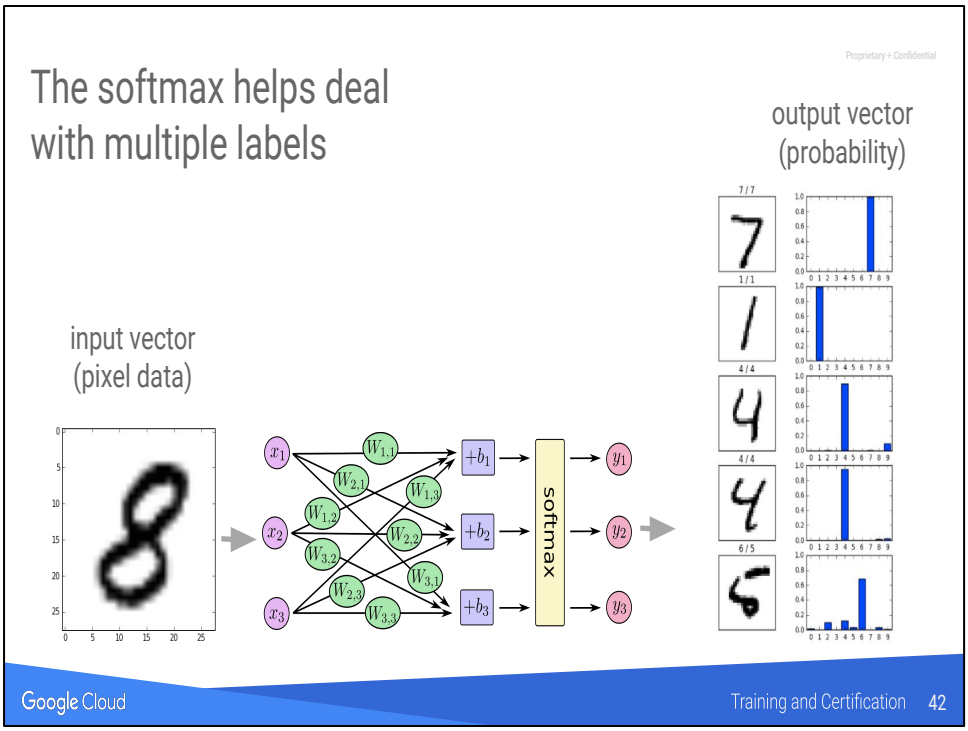
Each pixel value is an input



28 x 28 gray scale image =

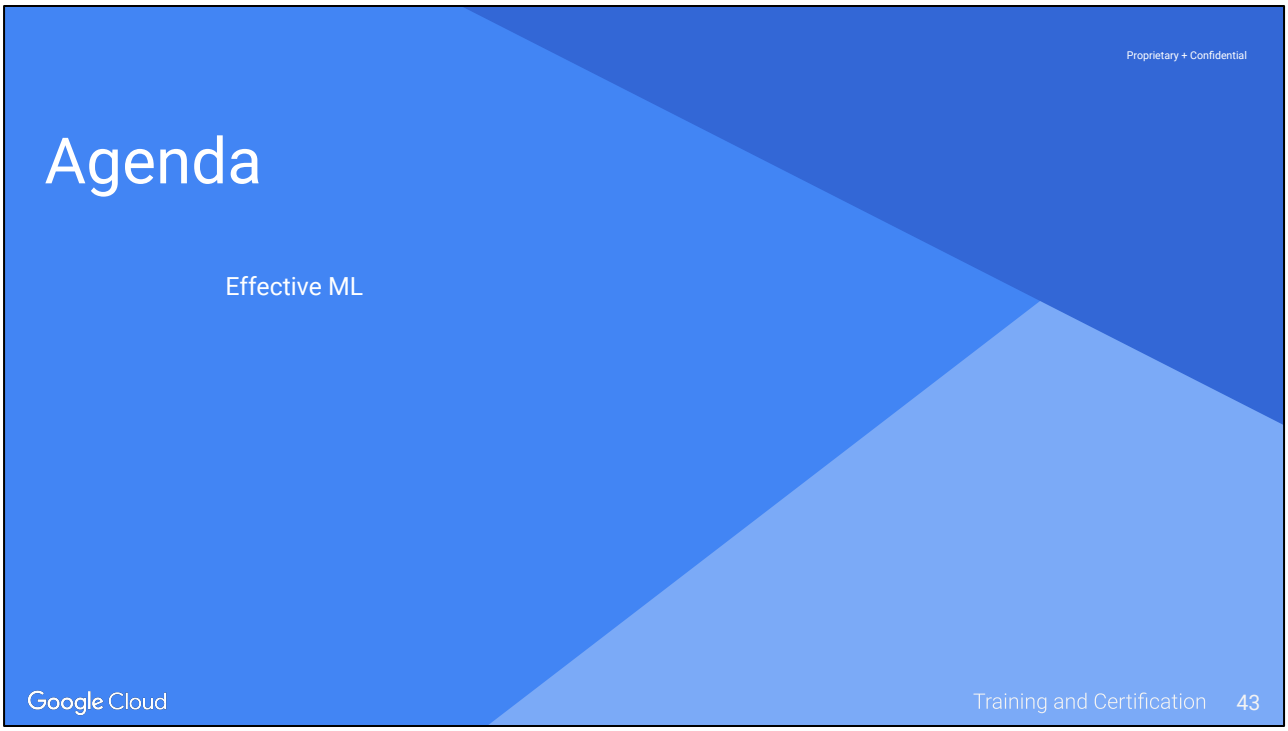
784 numbers

$$\sum_{i}^{n} w_i x_i > b$$



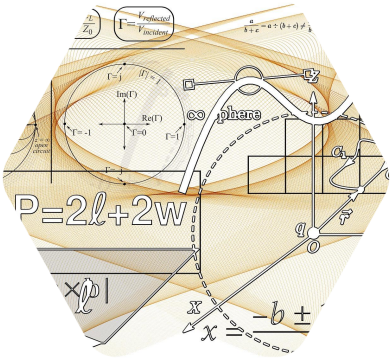
Notes:

The softmax essentially normalizes the labels so that total probability is 1.0. It also emphasizes the peaks more than just a simple divide by sum.



20-40 minutes based on audience.

The popular imagination of what ML is



Lots of data

Complex mathematics in multidimensional spaces

Magical results

Notes:

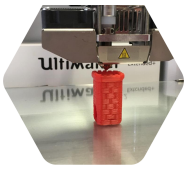
https://pixabay.com/en/x-y-mathematics-equation-937883/ (cc0)
https://pixabay.com/en/mathematics-formula-physics-school-1233876/ (cc0)
https://pixabay.com/en/deer-statue-silhouette-animal-stag-732122/ (cc0)

In reality, ML is...

Collect data

Organize data

Create model



Use machines to flesh out the model from data

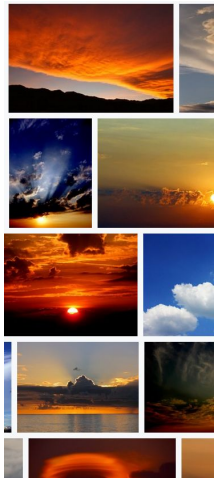
Deploy fleshed out model

Notes:

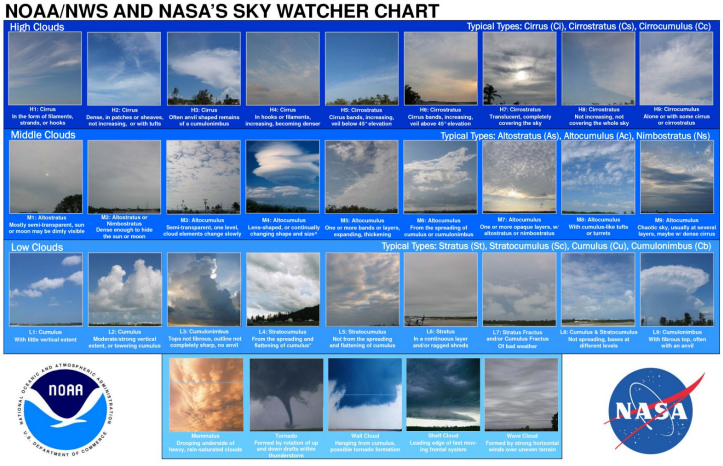
https://pixabay.com/en/ant-brown-carrying-egg-white-44588/ (cc0)
https://pixabay.com/en/tile-organization-exterior-materials-846016/ (cc0)
https://pixabay.com/en/deer-dream-animal-fantasy-1333814/ (cc0), cropped
https://pixabay.com/en/printer-3d-pressure-3d-printing-1455169/ (cc0)
https://pixabay.com/en/deer-statue-silhouette-animal-stag-732122/ (cc0)

On GCP, we can use:
Logging APIs, Cloud Pub/Sub, etc. and other real-time streaming to collect the data.
BigQuery, Dataflow and ML preprocessing SDK to organize the data [different types of organization].
TensorFlow to create the model.
Cloud ML to train, deploy the model.
The magic is still there …

The Dataset should cover all cases



All types covered?



Notes:

If you are going to recognize clouds, make sure you know what all types of clouds there are, and collect enough images of each type. Domain experts have to be involved in creating the dataset for ML. But this isn't enough … (see next slide)

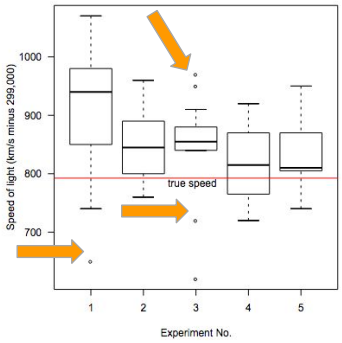
Image source:
The left-hand side is a snapshot of several cc0 images from Pixabay.
Right-hand side is from NOAA/NWS and NASA. so public domain.

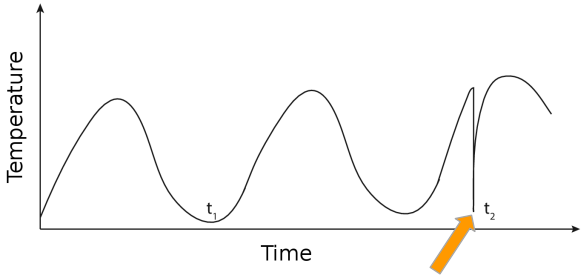
Negative examples and near-misses

Notes:

https://pixabay.com/en/clouds-sky-blue-cumulus-white-34027/ (cc0) cartoon cloud
https://pixabay.com/en/sheep-agriculture-animals-17482/ (cc0) flock of sheep
https://pixabay.com/en/scheuchzers-cottongrass-175409/ (cc0) field of cotton
https://pixabay.com/en/marble-background-backdrop-1006628/ (cc0) floor tile texture is cloud-like
https://pixabay.com/en/rocket-launch-steam-smoke-trail-693270/ (cc0) is a steam from a rocket
https://pixabay.com/en/chemtrail-conspiracy-theory-contrail-822000/ (cc0) is a chemtrail from plane

Explore the data you have; fix problems



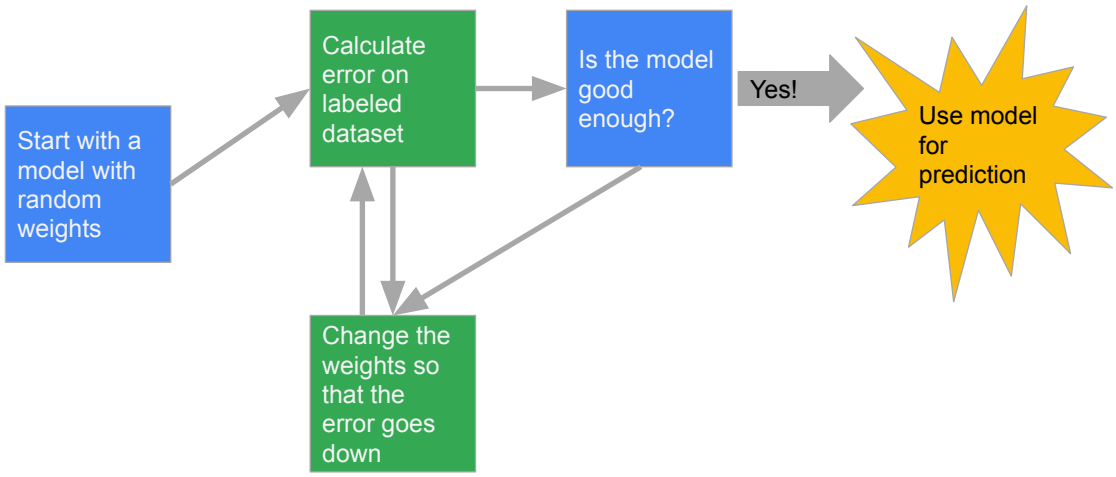


Notes:

Figure out what happened at the marked data points. It might be instrument errors, or worse.

Images from wikipedia page on outliers.

How do we compute error metrics?



Start from the outcome for a single example

ML system says you'll win $89.

Truth: You lose $14.

Notes:

Image: https://pixabay.com/en/gallop-horse-horse-racing-1117183/

Regression error: How far from true outcome?

ML system says you'll win $89.
Truth: You lose $14.
Your outcome is the error.

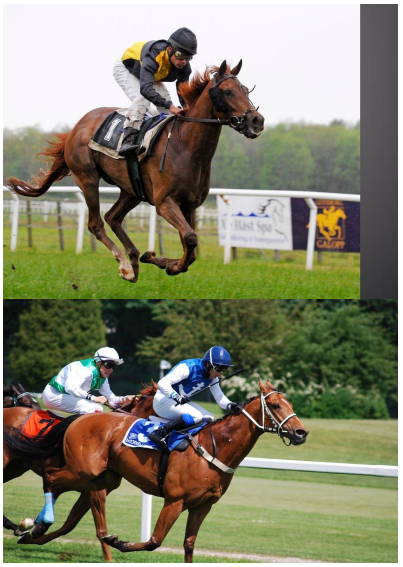
Error = truth - prediction
= -$14 - $89 = -$103

Notes:

Image: https://pixabay.com/en/gallop-horse-horse-racing-1117183/

Calculating regression error—Outcomes

-$103
+$75
-$10
+$99
-$113
+$82
+$56



Notes:

Top Right Image: https://pixabay.com/en/gallop-horse-horse-racing-1117183/
Bottom Right Image: https://pixabay.com/en/race-horses-racecourse-744105/

Calculating regression error

1) Get the error:
-$103
+$75
-$10
+$99
-$113
+$82
+$56

2) Square the error:
10609
5625
100
9801
12769
6724
3136

3) Calculate the mean:
6966

MSE
mean squared error

Notes:

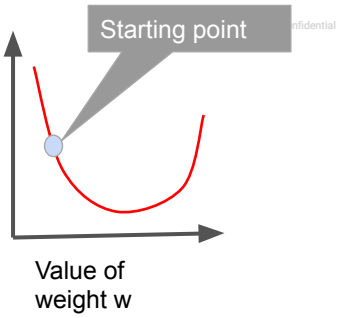
Image: https://pixabay.com/en/gallop-horse-horse-racing-1117183/

Mathematically...

Y-cap is the model estimate
Y is the labeled value
Mean Square Error (MSE) is:

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

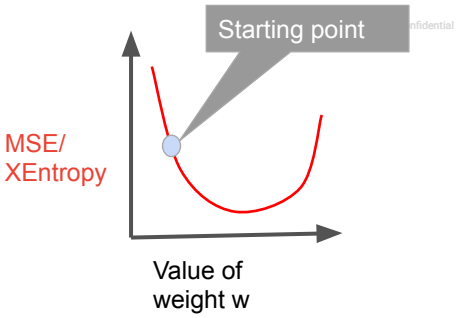
MSE



Notes:

Mathematically speaking, we prefer differentiable error measures so that we can do gradient descent. Mean-square error is differentiable.

For classification problems, we use cross-entropy



For classification problems, the most commonly used error measure is *cross-entropy*—because it is differentiable:

$$-\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

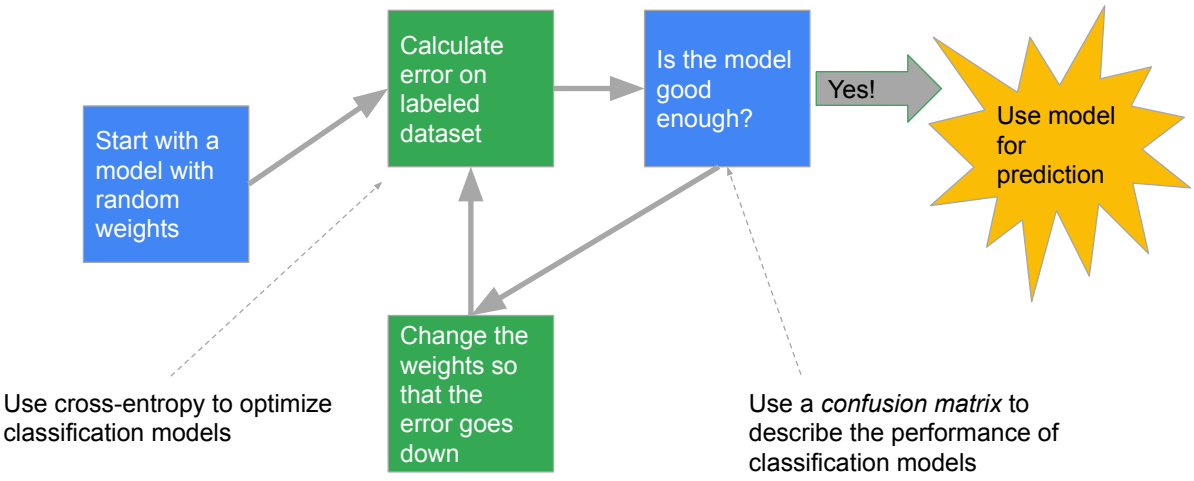
Notes:

Mathematically speaking, we prefer differentiable error measures so that we can do gradient descent. So, for classification, we'll use cross-entropy.

Why cross-entropy for categorical variables? See: https://en.wikipedia.org/wiki/Cross_entropy#Cross-entropy_error_function_and_logistic_regression.

In non-mathematical terms: You use a neural network whose output node is a soft-max (so that outputs are restricted to [0,1]) and whose objective function is cross-entropy. The result of the NN can be treated as a probability. So if you train using data on whether a customer bought your product or did not buy the product and you train on that data using the right NN structure, the trained NN can be used to estimate the probability that a visitor to your website will buy your product! This is hugely useful.

Cross-entropy is not intuitive to business users



Use cross-entropy to optimize classification models

Use a *confusion matrix* to describe the performance of classification models

Confusion matrix

1) Get the outcomes

TN

FP

TP

FN

TP

FN

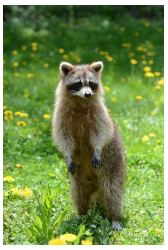
		ML System Says	
		Cat	No Cat
Truth	Cat	True Positive #TP	False Negative #FN
	No Cat	False Positive #FP	True Negative #TN

Notes:

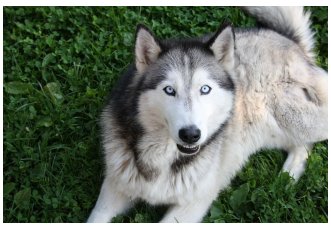
Get the outcome for each image in dataset, then add them up
Image: clipart https://pixabay.com/en/cartoon-cat-cute-1292872/
Confusion matrix is great, but it is 4 numbers and business decision makers want to see only one. Which one?

Accuracy, precision, and recall

Classify the cats!





Notes:

Image: cat with shoes courtesy of course author, Cassie Kozyrkov
Images: https://pixabay.com/en/cat-cat-face-sleep-exhausted-1551810/
https://pixabay.com/en/cat-pet-mirror-697113/
https://pixabay.com/en/tiger-sumatran-sumatran-tiger-164905/
https://pixabay.com/en/cat-wink-funny-fur-animal-red-1333926/
https://pixabay.com/en/huskies-dog-animal-blue-eyes-view-1370230/
https://pixabay.com/en/goldhamster-hamster-animal-nuts-943373/
https://pixabay.com/en/racoon-animal-garden-summer-1453600/

Hypothetical results from ML

Notes:

Image: cat with shoes courtesy of course author, Cassie Kozyrkov
Images: https://pixabay.com/en/cat-cat-face-sleep-exhausted-1551810/
https://pixabay.com/en/cat-pet-mirror-697113/
https://pixabay.com/en/tiger-sumatran-sumatran-tiger-164905/
https://pixabay.com/en/cat-wink-funny-fur-animal-red-1333926/
https://pixabay.com/en/huskies-dog-animal-blue-eyes-view-1370230/
https://pixabay.com/en/goldhamster-hamster-animal-nuts-943373/
https://pixabay.com/en/racoon-animal-garden-summer-1453600/
https://pixabay.com/en/cartoon-cat-cute-1292872/

Accuracy is fraction correct

Accuracy = 3 / 8
= 0.375

Notes:

Image: cat with shoes courtesy of course author, Cassie Kozyrkov
Images: https://pixabay.com/en/cat-cat-face-sleep-exhausted-1551810/
https://pixabay.com/en/cat-pet-mirror-697113/
https://pixabay.com/en/tiger-sumatran-sumatran-tiger-164905/
https://pixabay.com/en/cat-wink-funny-fur-animal-red-1333926/
https://pixabay.com/en/huskies-dog-animal-blue-eyes-view-1370230/
https://pixabay.com/en/goldhamster-hamster-animal-nuts-943373/
https://pixabay.com/en/racoon-animal-garden-summer-1453600/
https://pixabay.com/en/cartoon-cat-cute-1292872/
https://pixabay.com/en/check-mark-tick-mark-check-correct-1292787/
https://pixabay.com/en/incorrect-delete-remove-cancel-red-294245/

Accuracy fails if dataset unbalanced

1000 parking spaces
990 of them are **taken**
10 are **available**

A ML model that identified only **one** of the ten **available** spaces:

Accuracy = 991/1000
= 0.991!

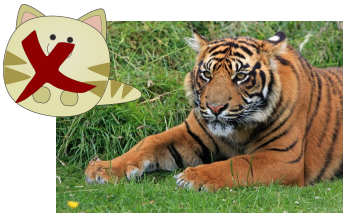
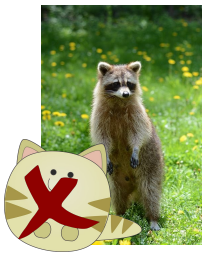
Notes:

Accuracy is fine for well-balanced datasets, but the dataset has a lot more of one category than another, we need to dig a little deeper into the performance. Use precision and recall for that. Suppose you have 1000 people and only 10 thieves. The ML can achieve 99.9% accuracy by classifying everything as “nice-people”. This is where it is helpful to know that recall=0.

https://pixabay.com/en/parking-autos-vehicles-traffic-825371/ (cc0)

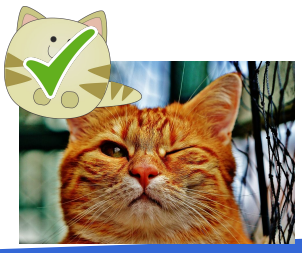
Precision = Positive Predictive Value

Accuracy when ML says “cat”


TP + FP = 5

Precision = TP / (TP + FP)
= 2 / 5 = 0.40



Notes:

Ask them what the precision is for the parking-lot example.
Answer: Depends on what you define as “positive”.
We have to define “available” as positive since that’s what we want to predict ...
Answer: Precision = 1/1 = 1 because all our predictions of “available” are correct.
When we are sure, we are sure.

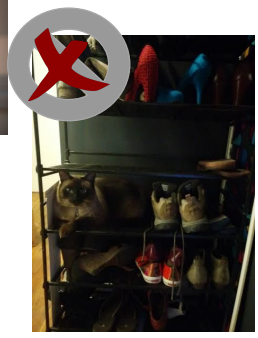
Image: cat with shoes courtesy Cassie Kozyrkov, Google
Images: https://pixabay.com/en/cat-cat-face-sleep-exhausted-1551810/
https://pixabay.com/en/cat-pet-mirror-697113/
https://pixabay.com/en/tiger-sumatran-sumatran-tiger-164905/
https://pixabay.com/en/cat-wink-funny-fur-animal-red-1333926/
https://pixabay.com/en/huskies-dog-animal-blue-eyes-view-1370230/
https://pixabay.com/en/goldhamster-hamster-animal-nuts-943373/
https://pixabay.com/en/racoon-animal-garden-summer-1453600/
https://pixabay.com/en/cartoon-cat-cute-1292872/
https://pixabay.com/en/check-mark-tick-mark-check-correct-1292787/
https://pixabay.com/en/incorrect-delete-remove-cancel-red-294245/

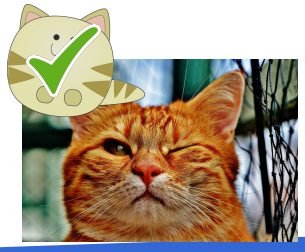
Recall is true positive rate

Recall = fraction of cats ML finds

TP + FN = 4

Recall = TP / (TP + FN)
= 2 / 4 = 0.50





Notes:

Ask them what the recall is for the parking-lot example.
Answer: Recall = 1/10 = 0.1

Image: cat with shoes courtesy Cassie Kozyrkov, Google
Images: https://pixabay.com/en/cat-cat-face-sleep-exhausted-1551810/
https://pixabay.com/en/cat-pet-mirror-697113/
https://pixabay.com/en/tiger-sumatran-sumatran-tiger-164905/
https://pixabay.com/en/cat-wink-funny-fur-animal-red-1333926/
https://pixabay.com/en/huskies-dog-animal-blue-eyes-view-1370230/
https://pixabay.com/en/goldhamster-hamster-animal-nuts-943373/
https://pixabay.com/en/racoon-animal-garden-summer-1453600/
https://pixabay.com/en/cartoon-cat-cute-1292872/
https://pixabay.com/en/check-mark-tick-mark-check-correct-1292787/
https://pixabay.com/en/incorrect-delete-remove-cancel-red-294245/

Do Now: In your own words, write down definitions for these ML terms

Term	Meaning
MSE	
Cross-entropy	
Accuracy	
Precision	
Recall	

Notes:

Mean Square Error: the loss measure for regression problems
Cross-entropy: the loss measure for classification problems
Accuracy: A more intuitive measure of skill for classifiers
Precision: Accuracy when classifier says “yes” (useful for unbalanced classes where there are many more yes-es than no-es)
Recall: Accuracy when the truth is “yes” (useful for unbalanced classes where there are very few yes-es)
ROC curve: a way to pick the threshold (of the probability that is output by the classifier) at which a specific precision or recall is reached. The area under the curve (AUC) is a threshold-independent measure of skill.

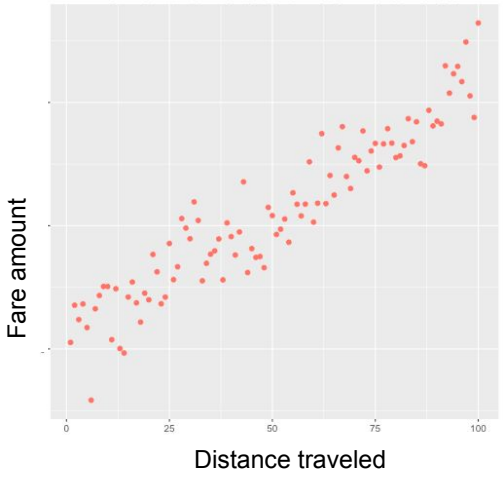
Agenda

Creating ML Datasets + Lab

Regression problem: Predict taxi fare

Problem: predict taxi fare amount based on distance traveled

What is the error measure to optimize?



Notes:

Simulation by Cassie Kozyrkov, Google.

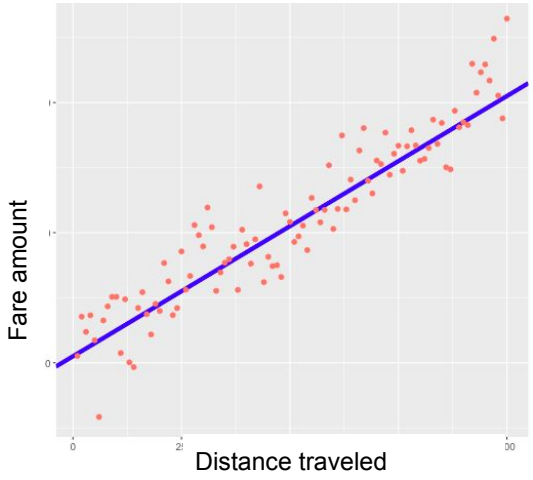
Regression problem, so typically MSE or RMSE.

Model 1

Red = training data

Blue = model prediction for each distance traveled

RMSE = 22.24



Notes:

Simulation by Cassie Kozyrkov, Google.

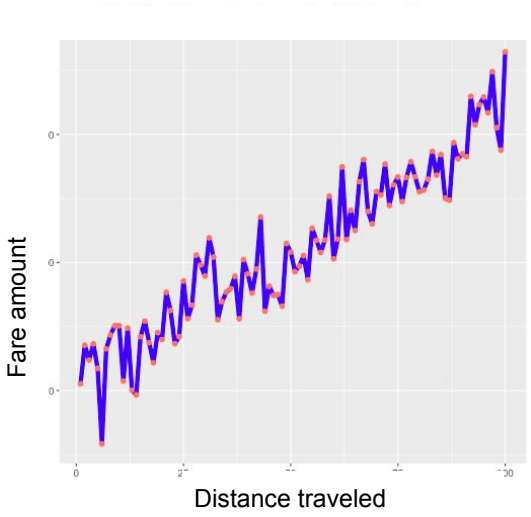
This is a linear model, so linear regression.

Model 2 has more free parameters

RMSE = 0

Which model is better?

How can we tell?



Notes:

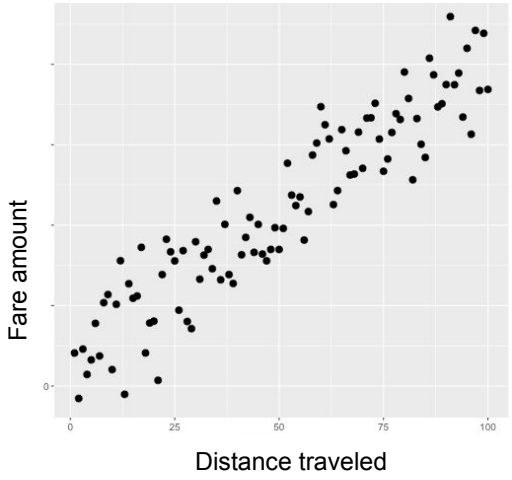
Simulation by Cassie Kozyrkov, Google.

This might be a polynomial of 100th order or a neural network with lots of nodes. A more complex model has more parameters that can be optimized. This can help is fit more complex data. But it might also help it memorize simpler datasets.

People intuitively feel that there is something fishy about Model 2.
But how can we tell? In ML, we often have lots of data, and no such intuition. Is a NN with 8 nodes better than a NN with 12 nodes? The 12 nodes has lower RMSE … so should we pick it? But what if we try 16 nodes, and it has even lower RMSE? At what point do we stop and say a model is now simply memorizing?

Does the model generalize to new data?

Need data that were not used in training



Notes:

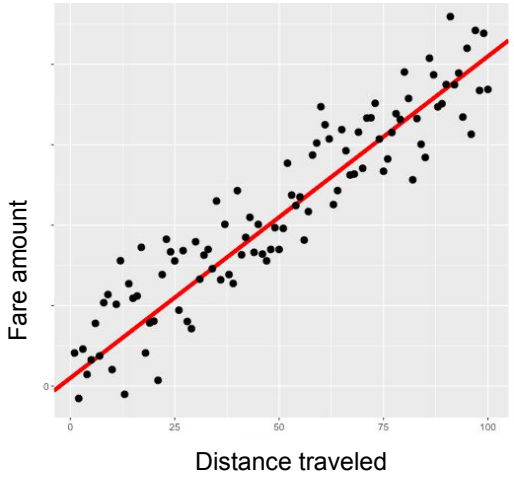
Simulation by Cassie Kozyrkov, Google.

Model 1 generalizes well

Old RMSE = 22.24

New RMSE = 21.98

Pretty similar = good



Notes:

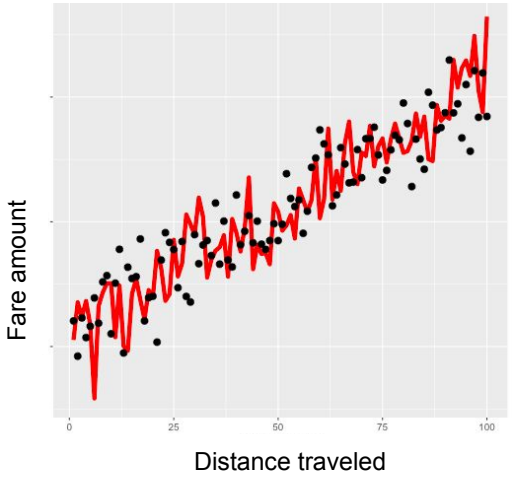
Simulation by Cassie Kozyrkov, Google.

Model 2 does not generalize well

Old RMSE = 0

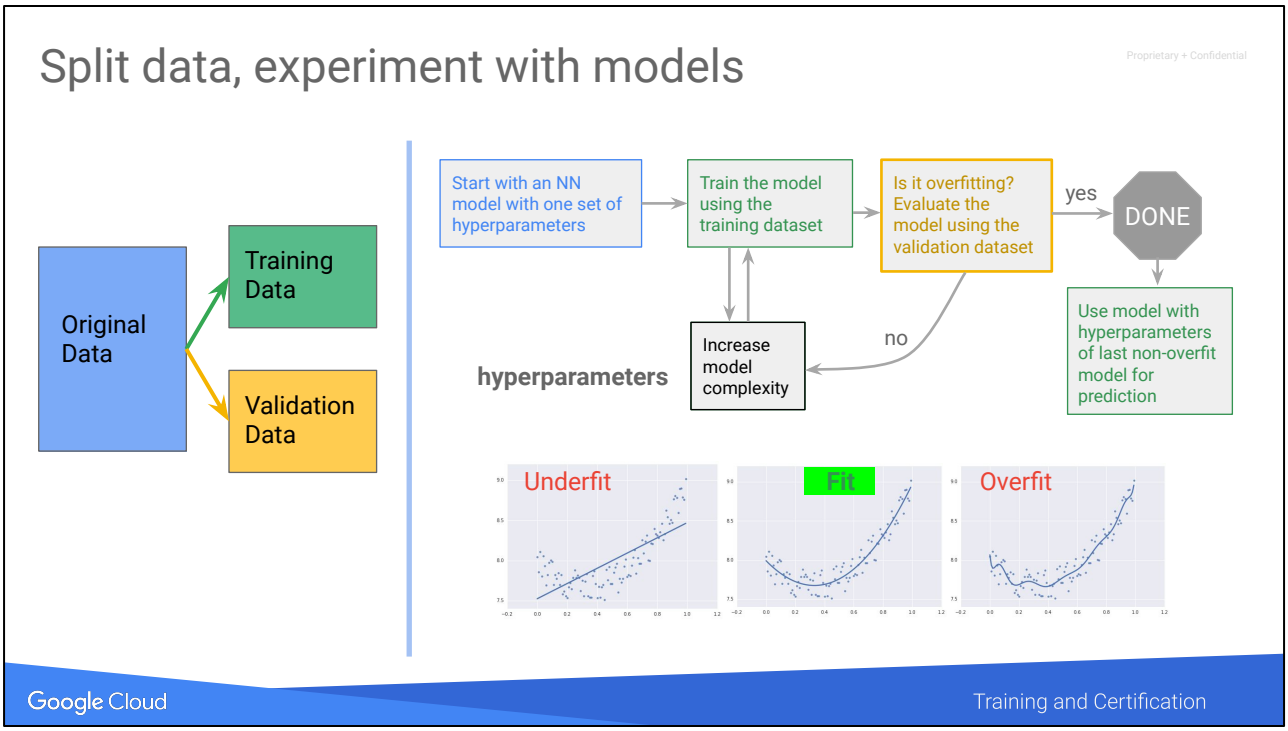
New RMSE = 32

This is a red flag



Notes:

Simulation by Cassie Kozyrkov, Google.



Training and evaluating an ML model is an experiment with finding the right generalizable model that fits your training dataset but doesn't memorize it. As you see here, we have an overly simplistic linear model that doesn't fit the relationships in the data. You'll be able to see how bad this is immediately by looking at your loss metric during training (and visually on this graph here as there are quite a few points outside the shape of trend line). This is called underfitting.

On the opposite end of the spectrum is overfitting, as shown on the right extreme. Here we greatly increased the complexity of our linear model and turned it into an n-th order polynomial which seems to model the training dataset really well -- almost too well. This is where the evaluation dataset comes in -- you can use the evaluation dataset to determine if the model parameters are leading to overfitting. Overfitting or memorizing your training dataset can be far worse than having a model that only adequately fits your data.

Somewhere in between an underfit where the loss metric is not low enough and an overfit whether the model doesn't generalize is the right level of model complexity.

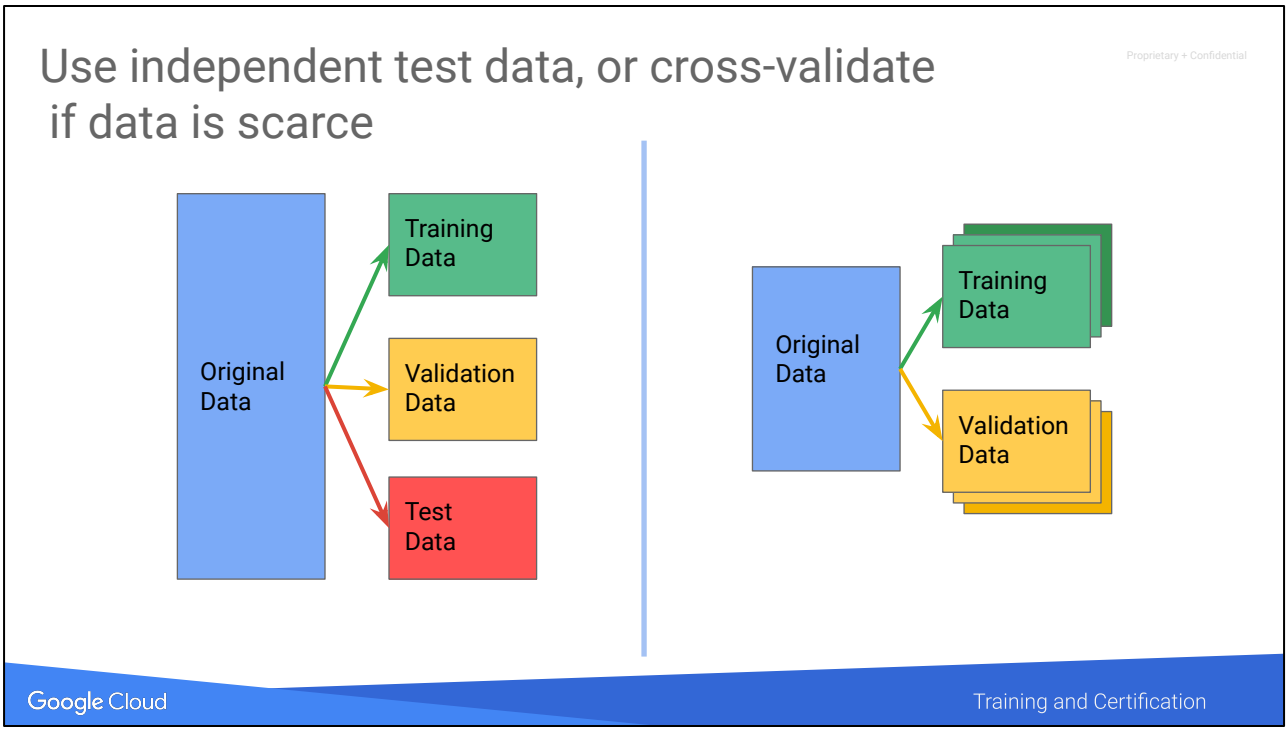
In addition to helping you choose between two completely different ML models (like regression or neural networks), you can also use your validation dataset to help fine-tune the hyperparameters of a single model which, if you recall, are set before training. This tuning process is accomplished through successive training runs and comparisons against your validation dataset to check for overfitting.

So here's how your validation dataset will actually be used after your model training. As you saw when we covered Optimization, training the model is where we start with random weights, calculate a loss metric on a batch of data, adjust the weights to minimize the loss metric, and repeat.

Periodically, we want to assess the performance of our model against data it has not yet seen in training which is where we use our validation dataset after a completed training run with the initial hyperparameters we set.

If there is not significant divergence between the loss metrics from the training run and the loss metric for the validation dataset then the model could still be optimized and tuned. We can go back to our hyperparameters that are set before training happens and try another training model run.
Perhaps we add one more layer to our neural network.

You can use a loop similar to this to also figure out model parameters like what we just did for hyperparameters, for example the number of layers or nodes to use in a neural network. Essentially, you will train with one configuration and evaluate on the validation dataset. Then, you will try out a different configuration that has more or fewer nodes and evaluate on the validation dataset. You will choose the model configuration that results in the lower loss on the validation dataset, not the model configuration that results in lower loss on the training one.

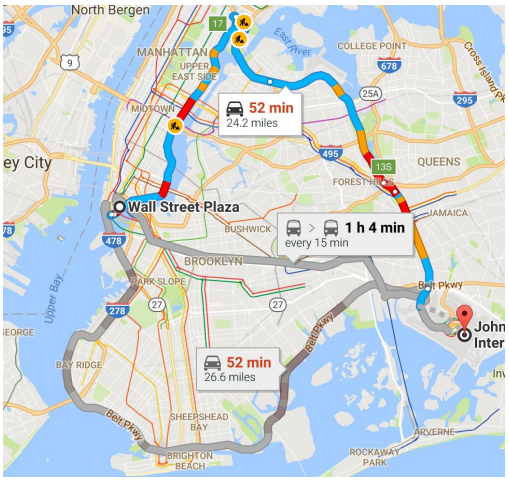


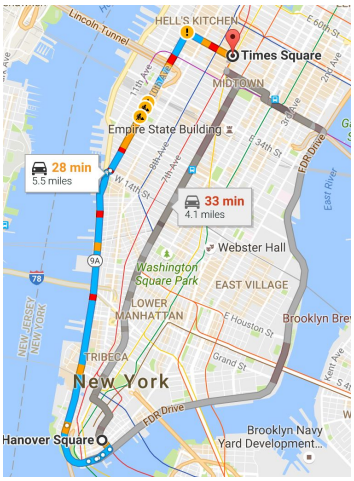
Notes

Cannot just use the original validation dataset. That is no longer truly independent.
Cross-validation could be done in both cases on the ice cream cone example. If we do random sampling, it's just a matter of choosing a different seed. If we divide by days/month, it's just a matter of varying the cutoff day (it was the first 25 days in our example, but we could cycle through and take the middle 25 days). This also ensures that we don't throw away any data completely due to leakage.

Included in Option 2 is the idea of going ahead and using all of our data and allowing the passage of time to get us an independent test data.

Goal: To estimate taxi fare





http://www.nyc.gov/html/tlc/html/passenger/taxicab_rate.shtml

Taxi fares:

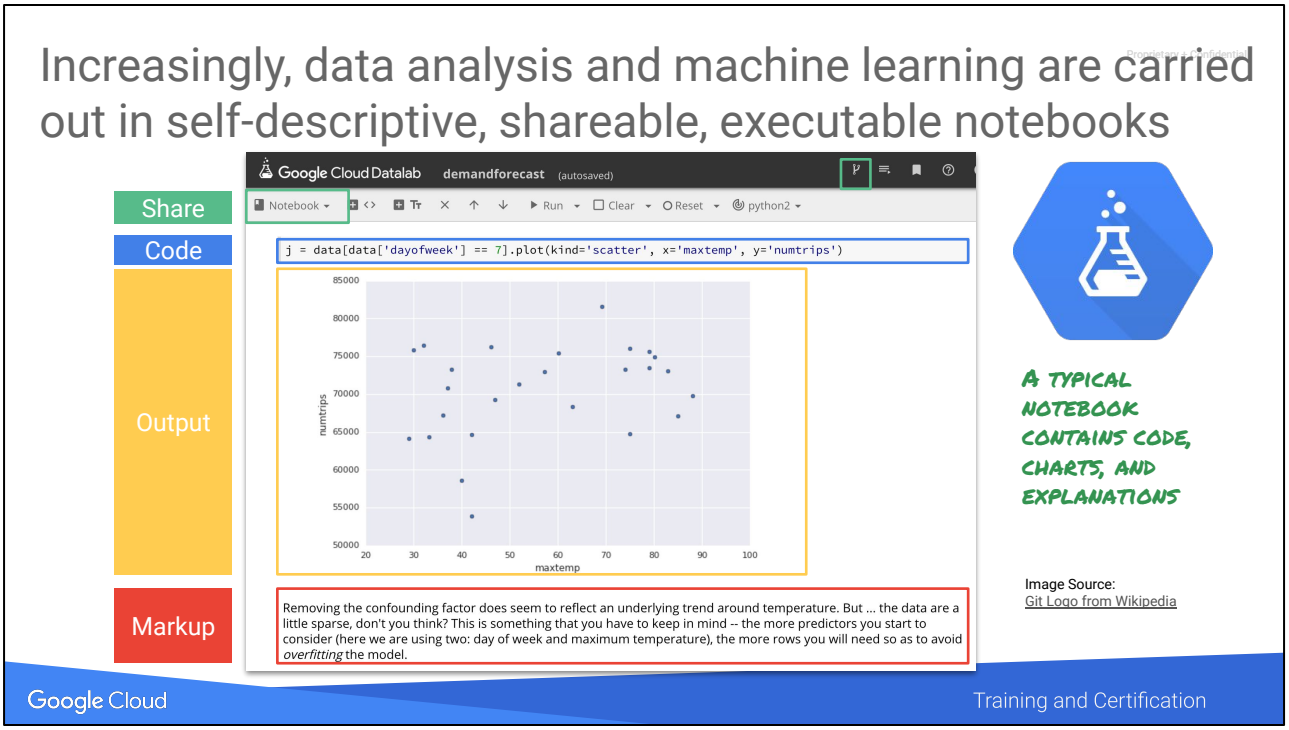
$2.50 initial charge
+
50c per ⅕ mile
(or)
50c per minute if stopped
+
Passenger pays tolls
+
Various special charges

Notes:

Left: a trip from Hanover Square to Time Square (downtown to midtown). Two routes -- the shorter one takes longer.

Right: a trip from JFK to Manhattan offers 2 routes, both very roundabout as compared to what a bus would take. You pay a toll to take the triborough bridge

But assume we don't know any of these things. Instead of hardcoding a bunch of rules, let's try to infer the fare amount simply from the data. The point of ML is to program with data instead of rules.
https://pixabay.com/en/question-mark-question-faq-ask-1421013/ (cc0)



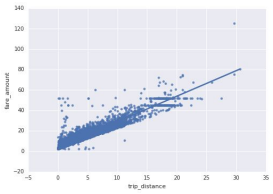
Do you use IPython or Jupyter notebooks today? Increasingly, data scientists work in self-descriptive, shareable, executable notebooks whenever they want to do data analysis or machine learning.
Datalab is based on Jupyter and it's open source.

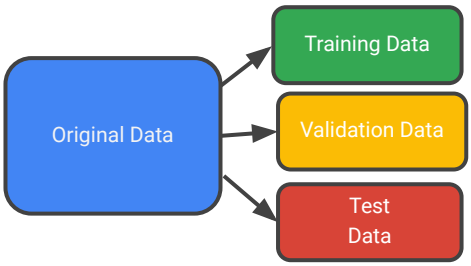
Combines code, documentation, results, and visualizations together in an intuitive notebook format
Cloud Datalab is built on Jupyter (formerly IPython)
Python, SQL, and JavaScript
Supports Google Charting or matplotlib for easy visualizations
How to launch:
From Cloud Shell: `datalab create <vm-name>`
Or Cloud Dataproc Initialization action

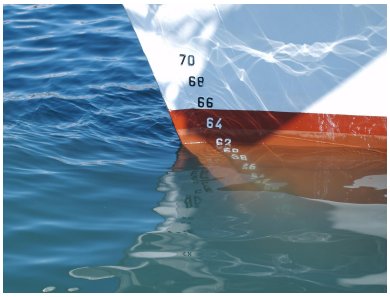
How to work with Cloud Datalab

1. Write code in Python
2. Run cell (Shift+Enter)
3. Examine output
4. Write commentary in markdown
5. Share and collaborate

Lab 1: Explore dataset, create ML datasets, create benchmark







1. Explore

2. Create Datasets

3. Benchmark

Notes:

These should form the first steps for any ML project that you undertake. You will often spend weeks just exploring the dataset to gain intuition into the problem -- this is crucial to creating good ML models. The benchmark phase should not be neglected; if you don't have a benchmark, you won't know what kind of performance you should seek to attain. Many times, errors can be detected simply by realizing that the performance of the ML model is nowhere near the benchmark.

https://pixabay.com/en/ship-sea-water-blue-water-level-1518522/ (cc0)


cloud.google.com
Image by Connie Zhou