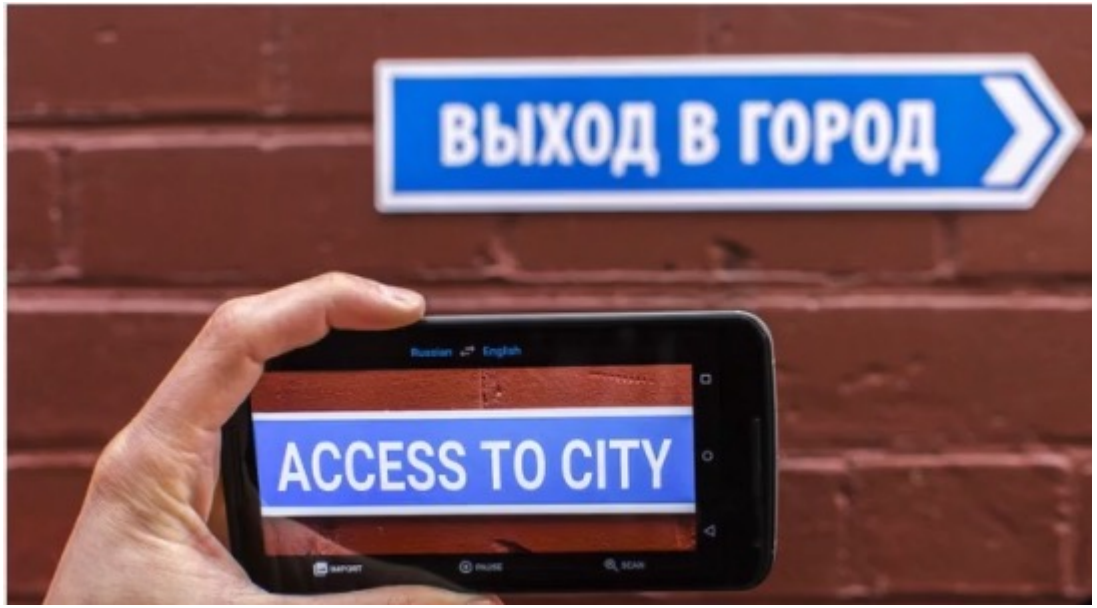
Deep Learning in Motion

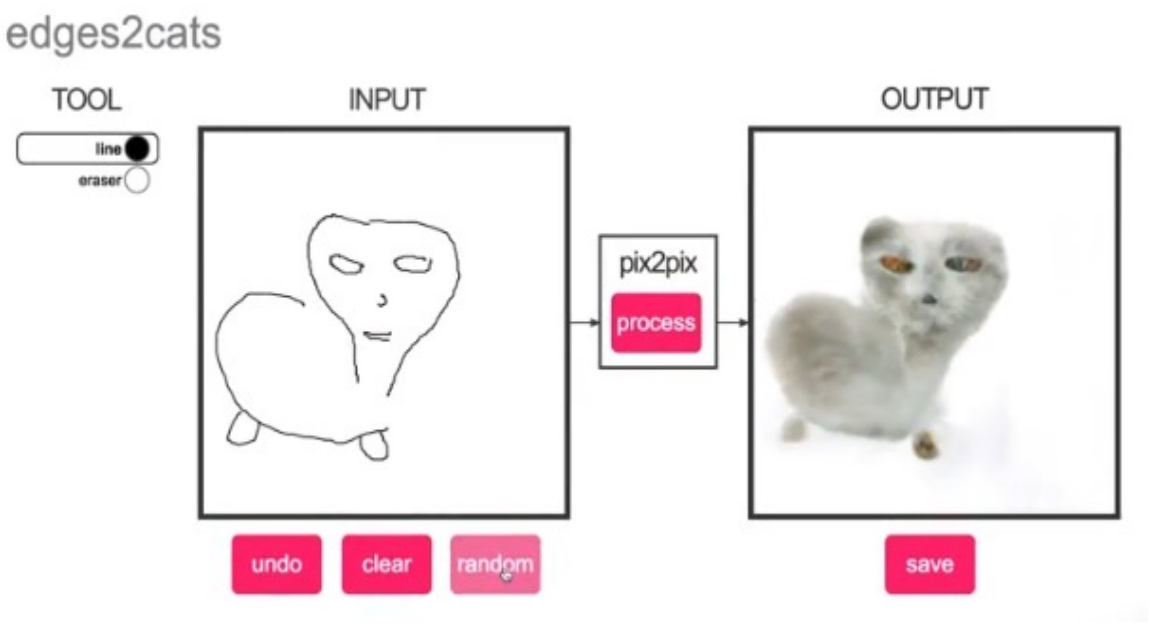
Unit 1 – Module 2
Getting Started

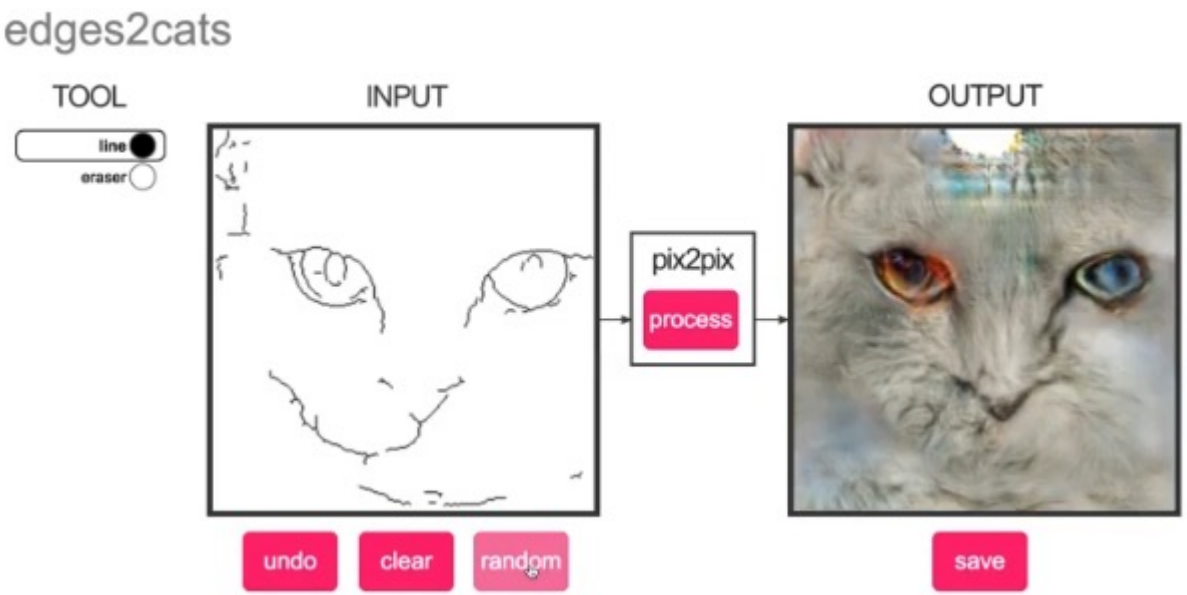
Beau Carnes

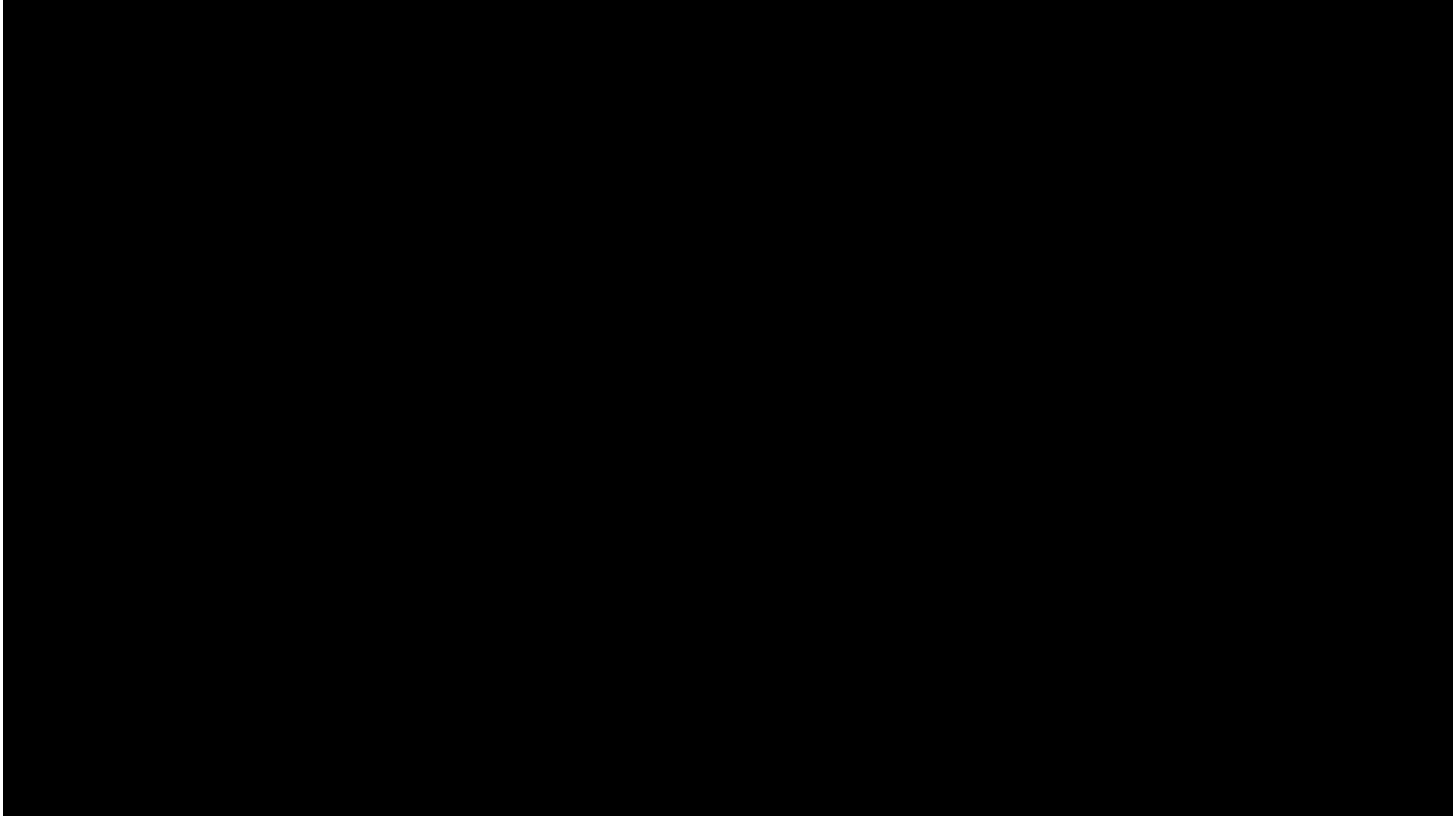
MANNING


ВЫХОД В ГОРОД
ACCESS TO CITY

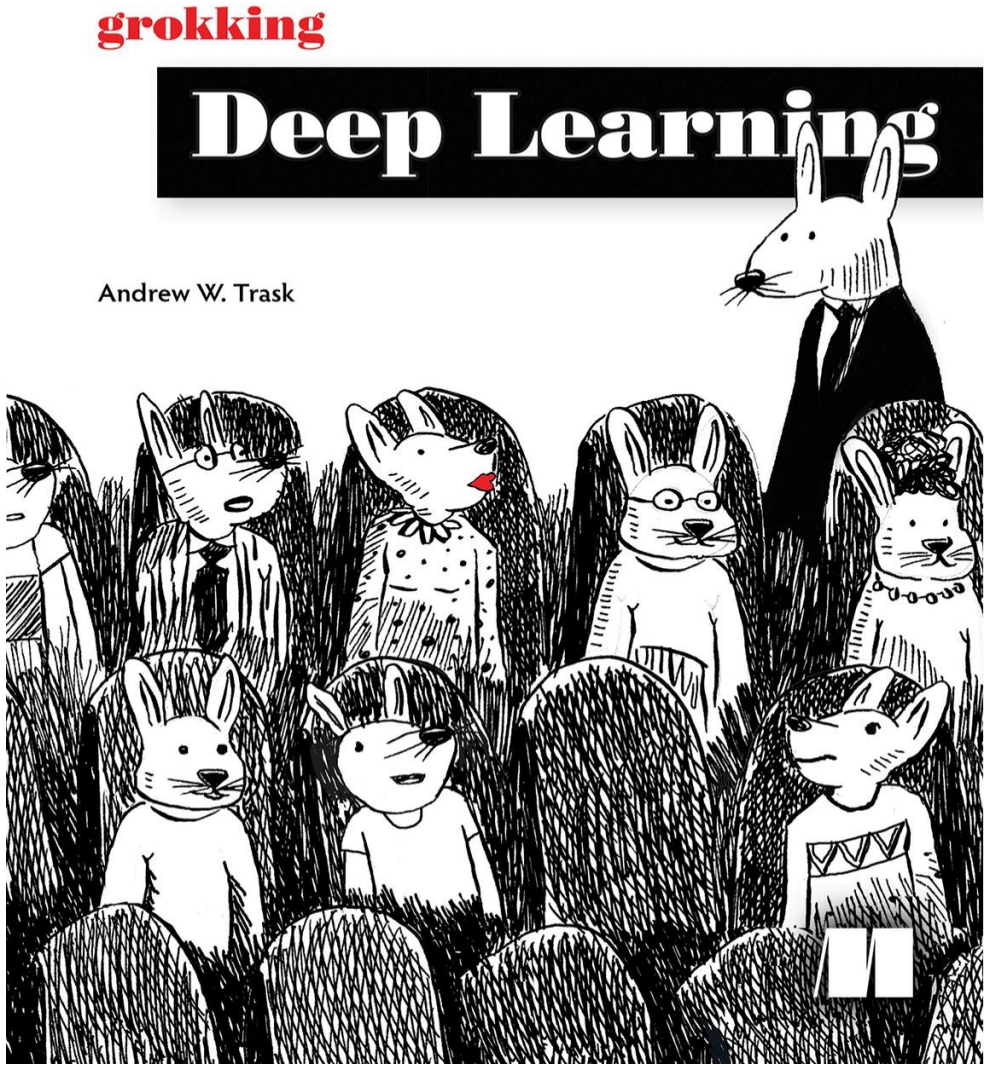

Google DeepMind
Challenge Match
8 - 15 March 2016
AlphaGo


edges2cats
TOOL
INPUT
OUTPUT
line
eraser
pix2pix
process
undo
clear
random
save


edges2cats
TOOL
INPUT
OUTPUT
line
eraser
pix2pix
process
undo
clear
random
save



https://www.youtube.com/watch?v=AmUC4m6w1wo



Google Scholar SIGN IN

Andrew Trask

FOLLOW

GET MY OWN PROFILE

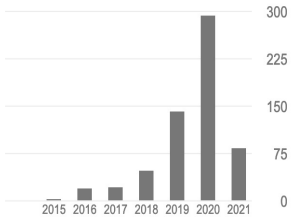
DeepMind, University of Oxford, and OpenMined
Verified email at google.com - Homepage

Deep Learning Differential Privacy Secure Multi-Party Comput… Federated Learning
Natural Language Processing

TITLE	CITED BY	YEAR
A generic framework for privacy preserving deep learning T Ryffel, A Trask, M Dahl, B Wagner, J Mancuso, D Rueckert, ... arXiv preprint arXiv:1811.04017	121	2018
sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings A Trask, P Michalak, J Liu arXiv preprint arXiv:1511.06388	121	2015
Neural arithmetic logic units A Trask, F Hill, S Reed, J Rae, C Dyer, P Blunsom arXiv preprint arXiv:1808.00508	105	2018
Systems and methods for neural language modeling A Trask, D Gilmore, M Russell US Patent 10.339.440	80	2019

Cited by

	All	Since 2016
Citations	629	620
h-index	9	9
i10-index	9	9



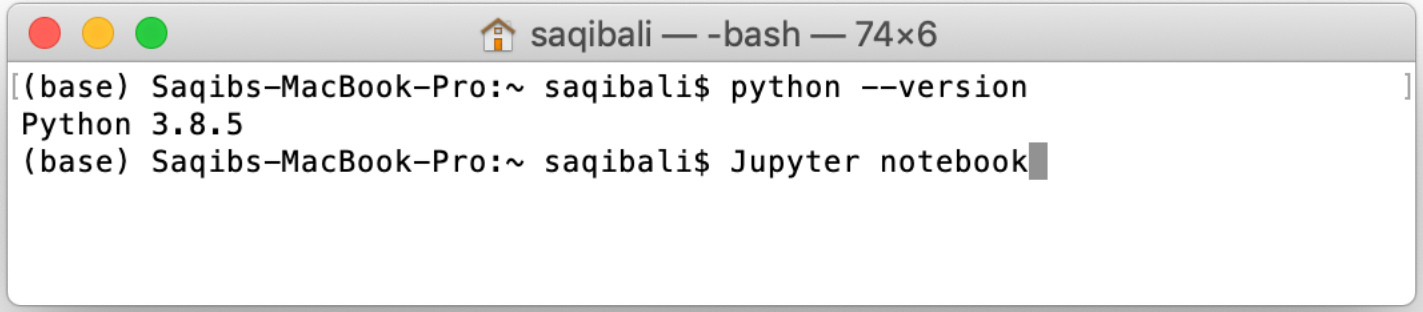
https://www.manning.com/books/grokking-deep-learning

https://github.com/iamtrask/Grokking-Deep-Learning

https://scholar.google.com/citations?user=2Ajxf1sAAAAJ&hl=en

Data science technology for groundbreaking research.

https://www.anaconda.com/


saqibali — -bash — 74×6
(base) Saqibs-MacBook-Pro:~ saqibali$ python --version
Python 3.8.5
(base) Saqibs-MacBook-Pro:~ saqibali$ Jupyter notebook


jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
Name
Last Modified
File size
Desktop 25 minutes ago
Documents 21 hours ago
Downloads 4 hours ago
Dropbox 38 minutes ago
Movies 4 years ago
Music 2 days ago
opt a day ago
Pictures 4 years ago
Public 4 years ago

jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

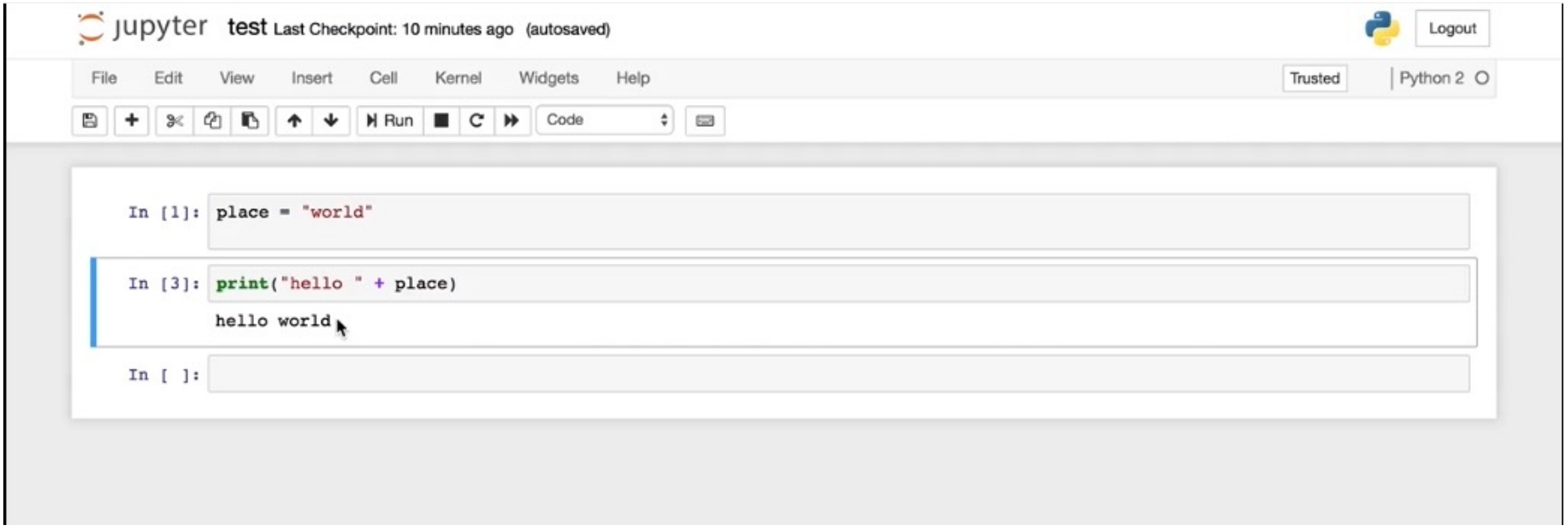
Upload New

Notebook:
Python 3

Other:
Text File
Folder
Terminal

0 / Name

Desktop	
Documents	
Downloads	
Dropbox	
Movies	4 years ago
Music	2 days ago
opt	a day ago
Pictures	4 years ago
Public	4 years ago



One of the most important parts of learning deep learning is the ability to stop a network while it's training and tear apart absolutely every piece to see what it looks like. This is something Jupyter Notebook is incredibly useful for.

What are Lists?

- Lists are used to store multiple items in a single variable.
- Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.
- Lists are created using square brackets:
- For Example:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

List Items

- List items are ordered, changeable, and allow duplicate values.
- List items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered

- When we say that lists are ordered, it means that the items have a defined order, and that order will not change.
- If you add new items to a list, the new items will be placed at the end of the list.

Changeable

- The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Allow Duplicates

- Since lists are indexed, lists can have items with the same value.
- **For Example -** Lists allow duplicate values:

```
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
```

List Length

- To determine how many items a list has, use the len() function.
- For Example - Print the number of items in the list.

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

List Items - Data Types

- List items can be of any data type.
- **For Example -** String, int and boolean data types.

```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```

A list with strings, integers and boolean values:

```
list1 = ["abc", 34, True, 40, "male"]
```

ObjectType

- **type()**
- From Python's perspective, lists are defined as objects with the data type 'list’.

```
<class 'list'>

What is the data type of a list?
mylist = ["apple", "banana", "cherry"]
print(type(mylist))
```

The list() Constructor

- It is also possible to use the list() constructor when creating a new list.
- **For Example -** Using the list() constructor to make a List:

```
thislist = list(("apple", "banana", "cherry")) # note the double
round-brackets
print(thislist)
```

2. Tuples

- Tuples are used to store multiple items in a single variable.
- A tuple is a collection which is ordered and **unchangeable**.
- Tuples are written with round brackets.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Tuple Items

- Tuple items are ordered, unchangeable, and allow duplicate values.
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Tuple Items

Tuple items are ordered, unchangeable, and allow duplicate values.

Ordered

When we say that tuples are ordered, it means that the items have a defined order, and that order will not change. Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Unchangeable

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Allow Duplicates

Since tuple are indexed, tuples can have items with the same value:

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

Tuple With One Item

To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Tuple Items - Data Types

- Tuple items can be of any data type:

String, int and boolean data types:
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)

- A tuple can contain different data types:

A tuple can contain different data types:
A tuple with strings, integers and boolean values:
tuple1 = ("abc", 34, True, 40, "male")

type()

- From Python's perspective, tuples are defined as objects with the data type 'tuple':

```
What is the data type of a tuple?
mytuple = ("apple", "banana", "cherry")
print(type(mytuple))
```

The tuple() Constructor

It is also possible to use the tuple() constructor to make a tuple.

Using the tuple() method to make a tuple:

```
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)
```

3. Sets

- Sets are used to store multiple items in a single variable.
- A set is a collection which is both *unordered* and *unindexed*.
- Sets are written with curly brackets.

Example
Create a Set:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Set Items

Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered

- Unordered means that the items in a set do not have a defined order.
- Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable

- Sets are unchangeable, meaning that we cannot change the items after the set has been created.
- Once a set is created, you cannot change its items, but you can add new items.
- **Duplicates Not Allowed**
- Sets cannot have two items with the same value. Duplicate values will be ignored:

```
thisset = {"apple", "banana", "cherry", "apple"}
print(thisset)
```

Length of a Set

- To determine how many items a set has, use the len() method.

Get the number of items in a set:

```
thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

Set Items - Data Types

- Set items can be of any data type:

String, int and boolean data types:
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}

- A set can contain different data types:

A set with strings, integers and boolean values:
set1 = {"abc", 34, True, 40, "male"}

type()

- From Python's perspective, sets are defined as objects with the data type 'set':

```
What is the data type of a set?
myset = {"apple", "banana", "cherry"}
print(type(myset))
```

The set() Constructor

- It is also possible to use the set() constructor to make a set.

```
Using the set() constructor to make a set:
thisset = set(("apple", "banana", "cherry")) # note the double round-brackets
print(thisset)
```

4. Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered, changeable and does not allow duplicates.
- As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.
- Dictionaries are written with curly brackets, and have keys and values:

Create and print a dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Dictionary Items

- Dictionary items are ordered, changeable, and does not allow duplicates.
- Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

Print the "brand" value of the dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```

Ordered or Unordered?

- As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.
- When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change.
- Unordered means that the items does not have a defined order, you cannot refer to an item by using an index.

Changeable

Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

Duplicates Not Allowed

Dictionaries cannot have two items with the same key:

Duplicate Values

Duplicate values will overwrite existing values:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

Dictionary Length

- To determine how many items a dictionary has, use the len() function:

Print the number of items in the dictionary:

```
print(len(thisdict))
```

Dictionary Items - Data Types

- The values in dictionary items can be of any data type:

String, int, boolean, and list data types:

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}
```

type()

- From Python's perspective, dictionaries are defined as objects with the data type 'dict':

Print the data type of a dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(type(thisdict))
```

Tuple Items

- Tuple items are ordered, unchangeable, and allow duplicate values.
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Tuple Items

- Tuple items are ordered, unchangeable, and allow duplicate values.
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

List	Tuples	Dictionary
A list is mutable	A tuple is immutable	A dictionary is mutable
Lists are dynamic	Tuples are fixed size in nature	In values can be of any data type and can repeat, keys must be of immutable type
List are enclosed in brackets[ ] and their elements and size can be changed	Tuples are enclosed in parenthesis ( ) and cannot be updated	Tuples are enclosed in curly braces { } and consist of key:value
Homogenous	Heterogeneous	Homogenous
Example: List = [10, 12, 15]	Example: Words = ("spam", "egss") Or Words = "spam", "eggs"	Example: Dict = {"ram": 26, "abi": 24}
Access: print(list[0])	Access: print(words[0])	Access: print(dict["ram"])
Can contain duplicate elements	Can contain duplicate elements. Faster compared to lists	Cant contain duplicate keys, but can contain duplicate values
Slicing can be done	Slicing can be done	Slicing can't be done
Usage: ❖ List is used if a collection of data that doesnt need random access. ❖ List is used when data can be modified frequently	Usage: ❖ Tuple can be used when data cannot be changed. ❖ A tuple is used in combination with a dictionary i.e.a tuple might represent a key.	Usage: ❖ Dictionary is used when a logical association between key:value pair. ❖ When in need of fast lookup for data, based on a custom key. ❖ Dictionary is used when data is being constantly modified.

NumPy Introduction

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.

Why Use NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
- Arrays are very frequently used in data science, where speed and resources are very important.

Why is NumPy Faster Than Lists?

- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.
- This behaviour is called locality of reference in computer science.
- This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

Which Language is NumPy written in?

- NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

Create a NumPy ndarray Object

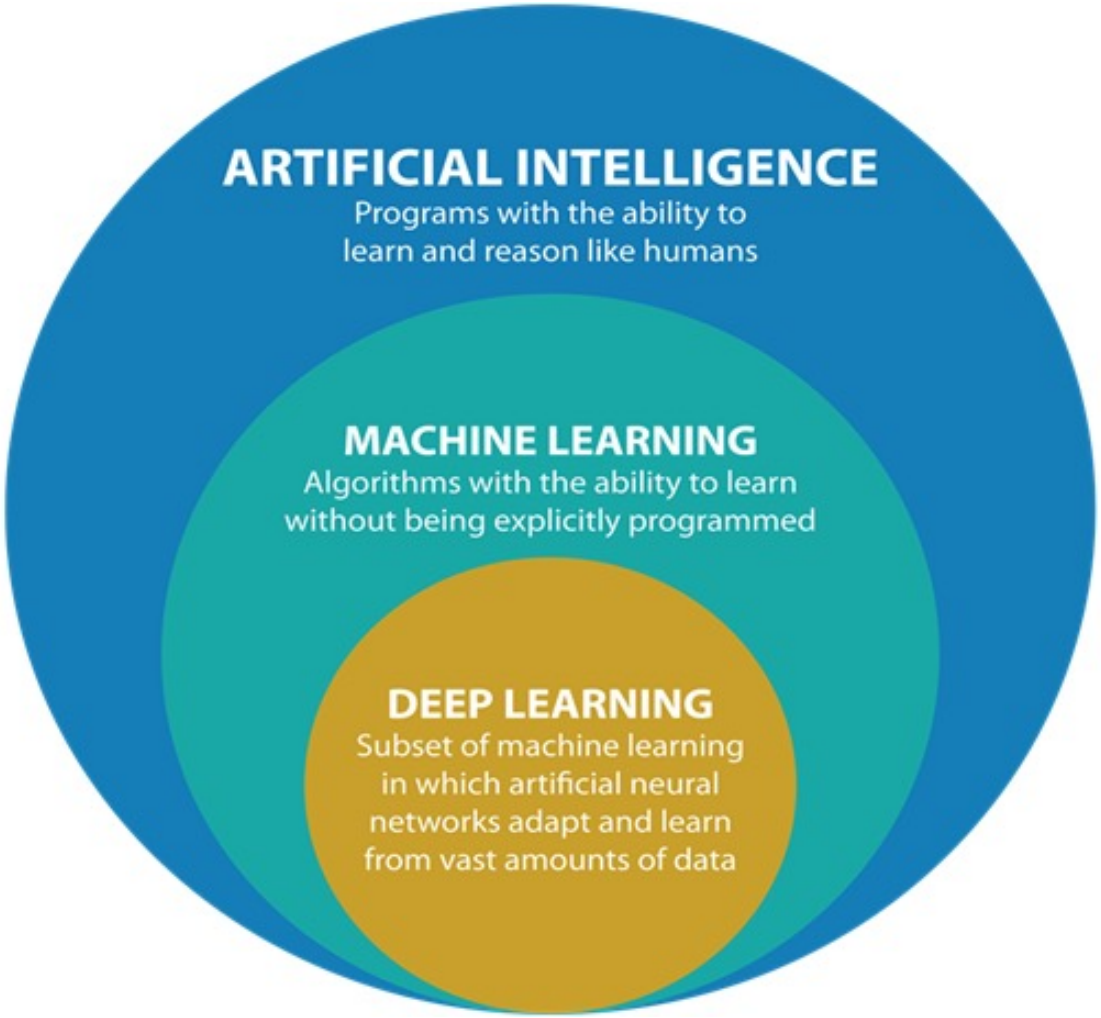
- NumPy is used to work with arrays. The array object in NumPy is called ndarray.
- We can create a NumPy ndarray object by using the array() function.
- For Example.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

To create an ndarray, we can pass a list, tuple or any array-like object into the array() method, and it will be converted into an ndarray:

```
import numpy as np
arr = np.array((1, 2, 3, 4, 5))
print(arr)
```

What is Deep Learning?



What is Machine Learning?

Machine learning is a subfield of computer science wherein *machines learn* to perform tasks for which they were *not explicitly programmed*. In short, machines observe a pattern and attempt to imitate it in some way that can be either direct or indirect.

Types of Machine Learning?

1. Supervised Learning learning is the direct imitation of a pattern between two datasets. It's always attempting to take an input dataset and transform it into an output dataset. For example.

- Using the **pixels** of an image to detect the *presence* or *absence of a cat*
- Using the **movies you've liked** to predict more *movies you may like*
- Using someone's **words** to predict whether they're *happy* or *sad*
- Using weather sensor **data** to predict the *probability of rain*
- Using car engine **sensors** to predict the optimal tuning *settings*
- Using news **data** to predict tomorrow's stock *price*
- Using an input **number** to predict a *number* double its size
- Using a raw **audio file** to predict a *transcript* of the audio

Supervised Learning

Supervised learning transforms datasets

Supervised machine learning is the direct imitation of a pattern between two datasets. It's always attempting to take an input dataset and transform it into an output dataset.

It's useful for taking *what you know* as input and quickly transforming it into *what you want to know*. The majority of work using machine learning results in the training of a supervised classifier of some kind.

Summary of Machine Learning
(Determining if an image contains a cat)

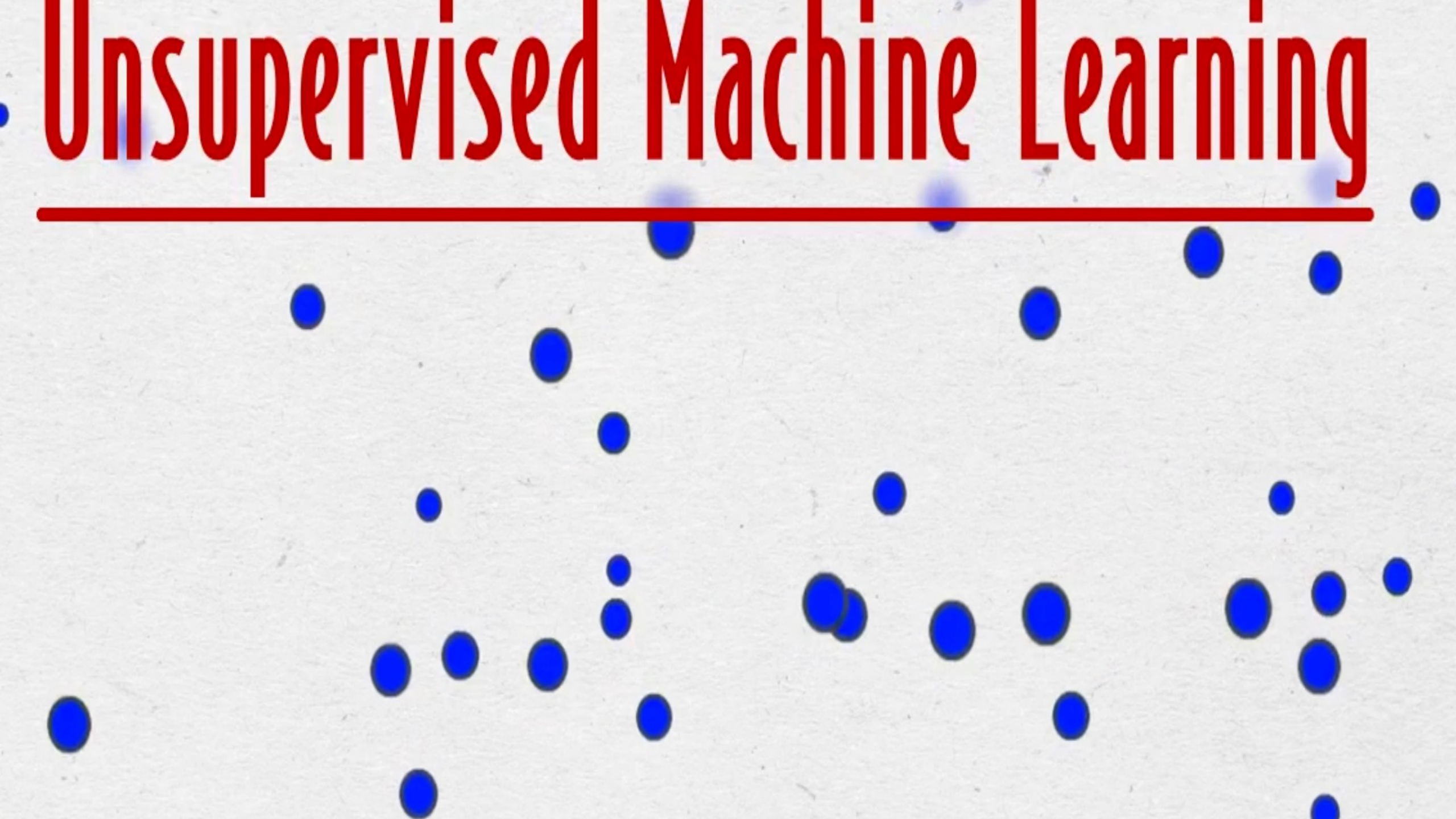
1. Feed the algorithm thousands of pictures - some with cats, some without
2. Tell the algorithm which pictures have cats and which do not
3. Algorithm extracts patterns from the data
4. Algorithm can hopefully correctly identify which pictures have cats

Unsupervised Learning

Unsupervised learning groups your data.

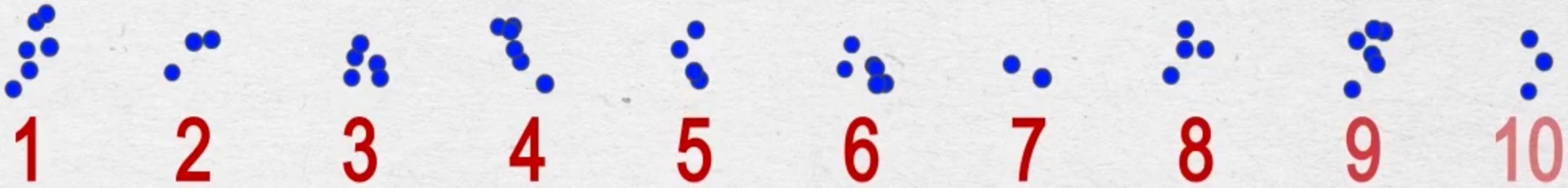
Unsupervised learning shares a property in common with supervised learning: it transforms one dataset into another. But the dataset that it transforms into is *not previously known or understood*. Unlike supervised learning, there is no “right answer” that you’re trying to get the model to duplicate. You just tell an unsupervised algorithm to “find patterns in this data and tell me about them.”

Unsupervised Machine Learning



Unsupervised Machine Learning

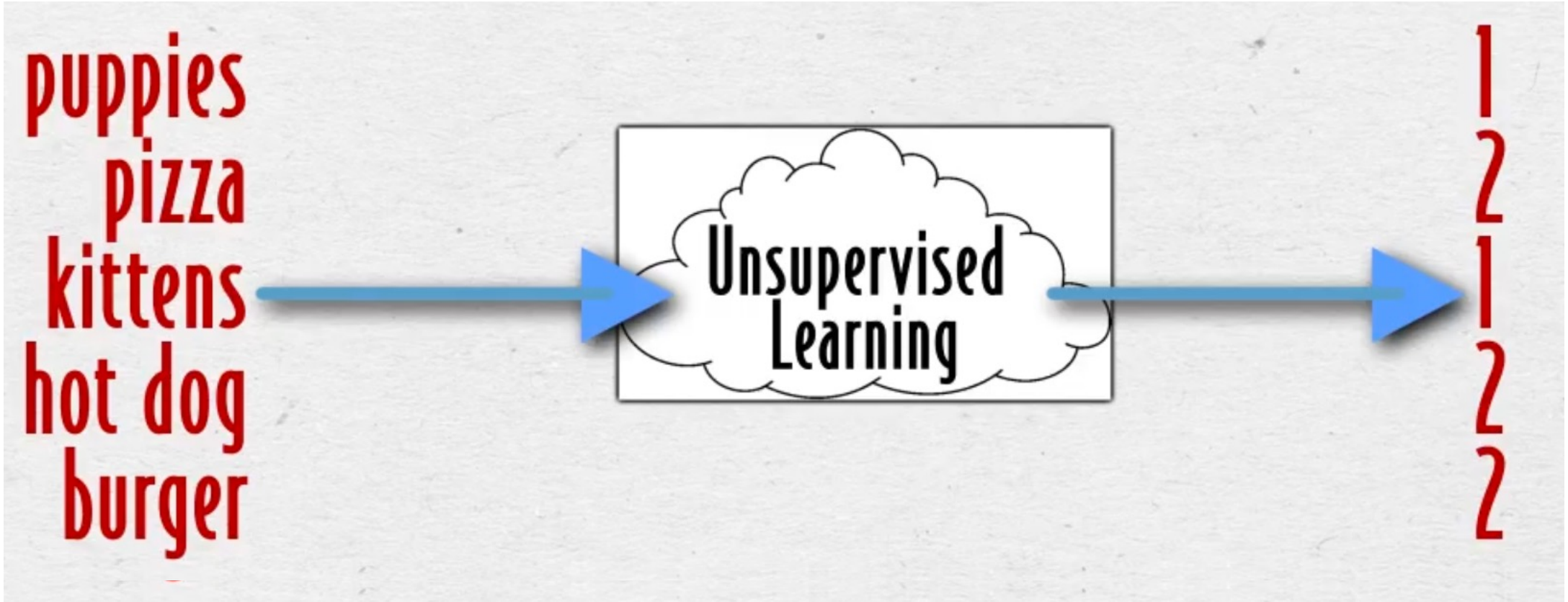
Clusters



Unsupervised Machine Learning

Unsupervised Learning

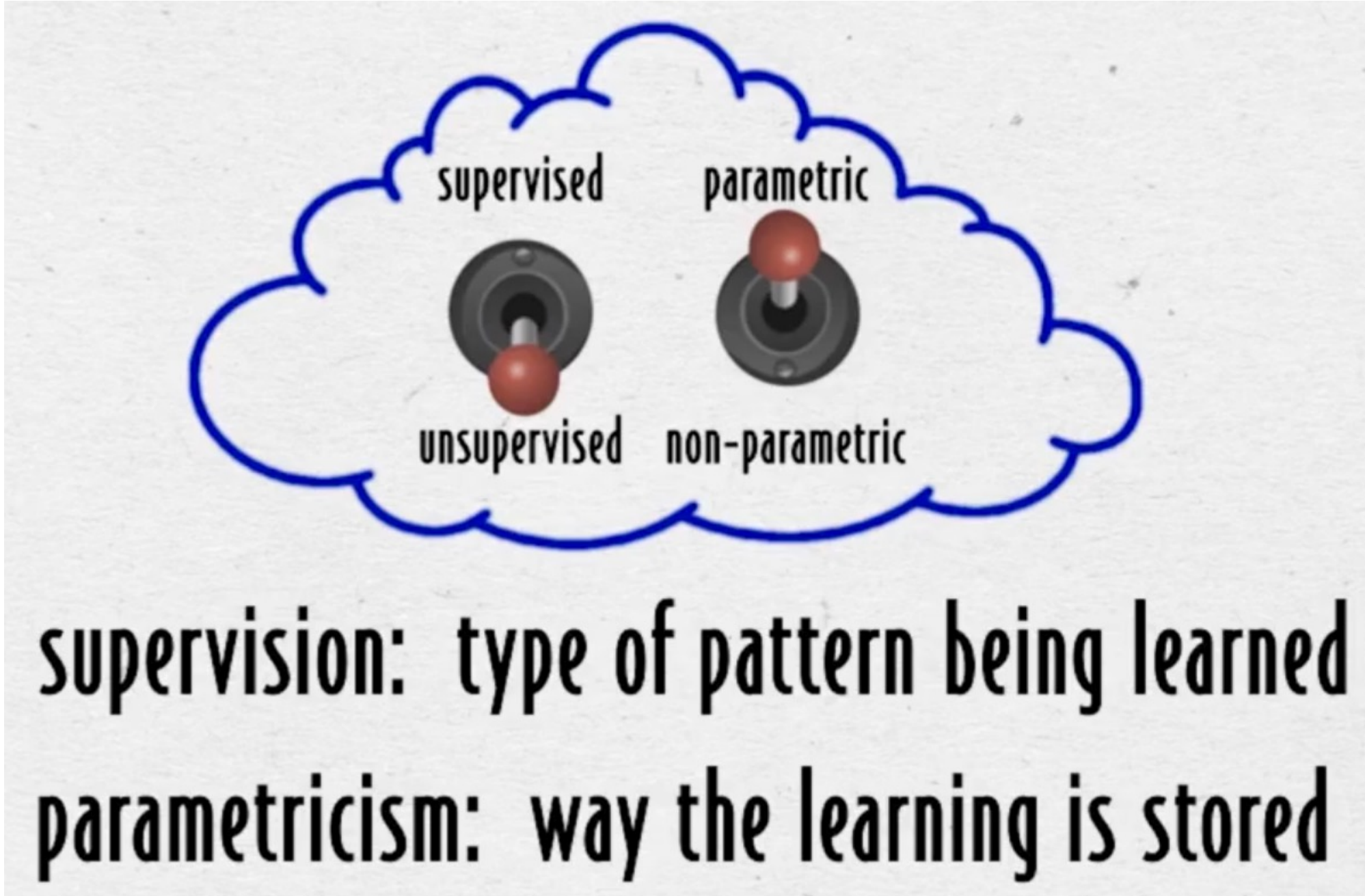
For example, clustering a dataset into groups is a type of unsupervised learning. Clustering transforms a sequence of datapoints into a sequence of cluster labels. If it learns 10 clusters, it's common for these labels to be the numbers 1–10.

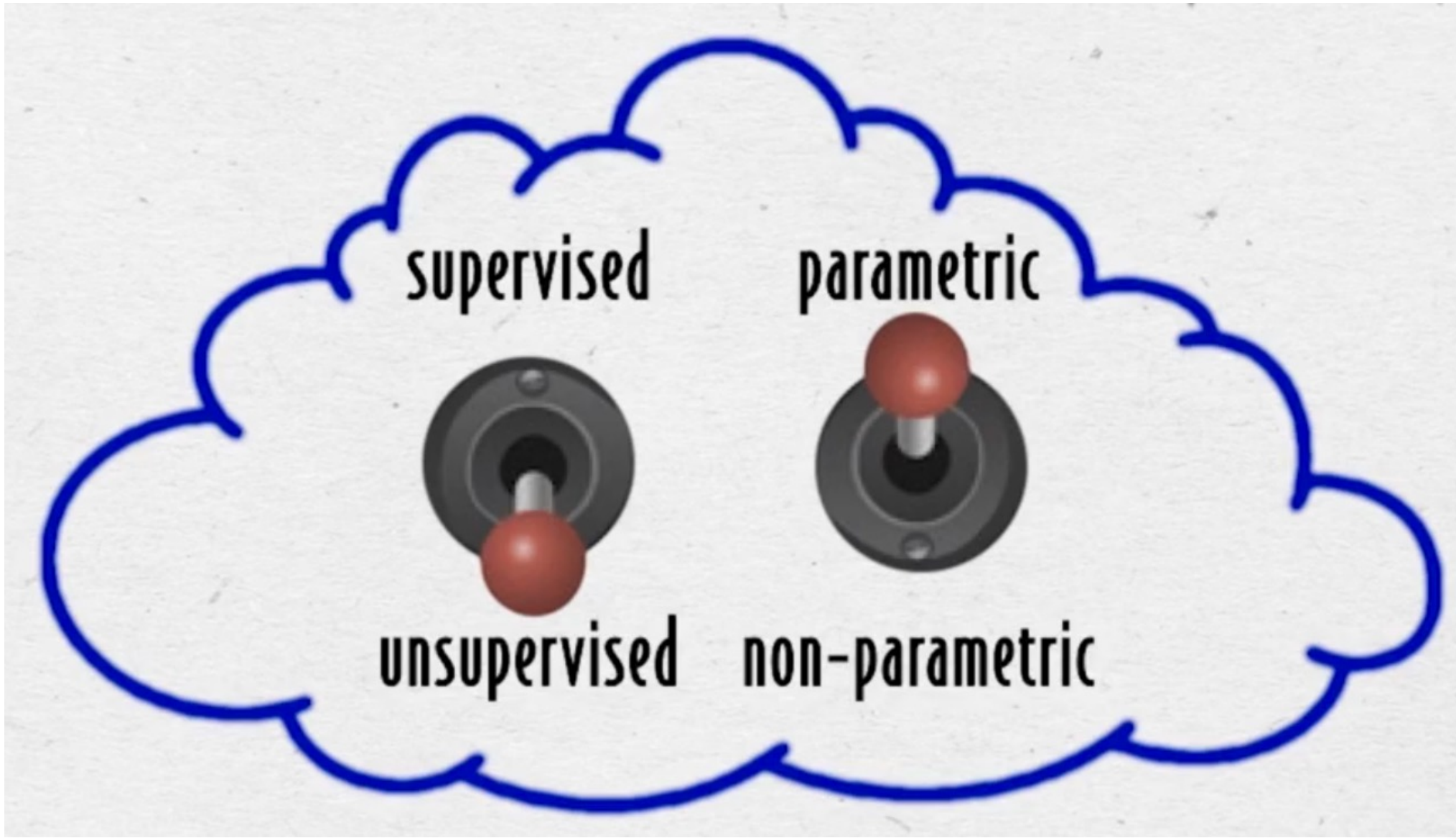


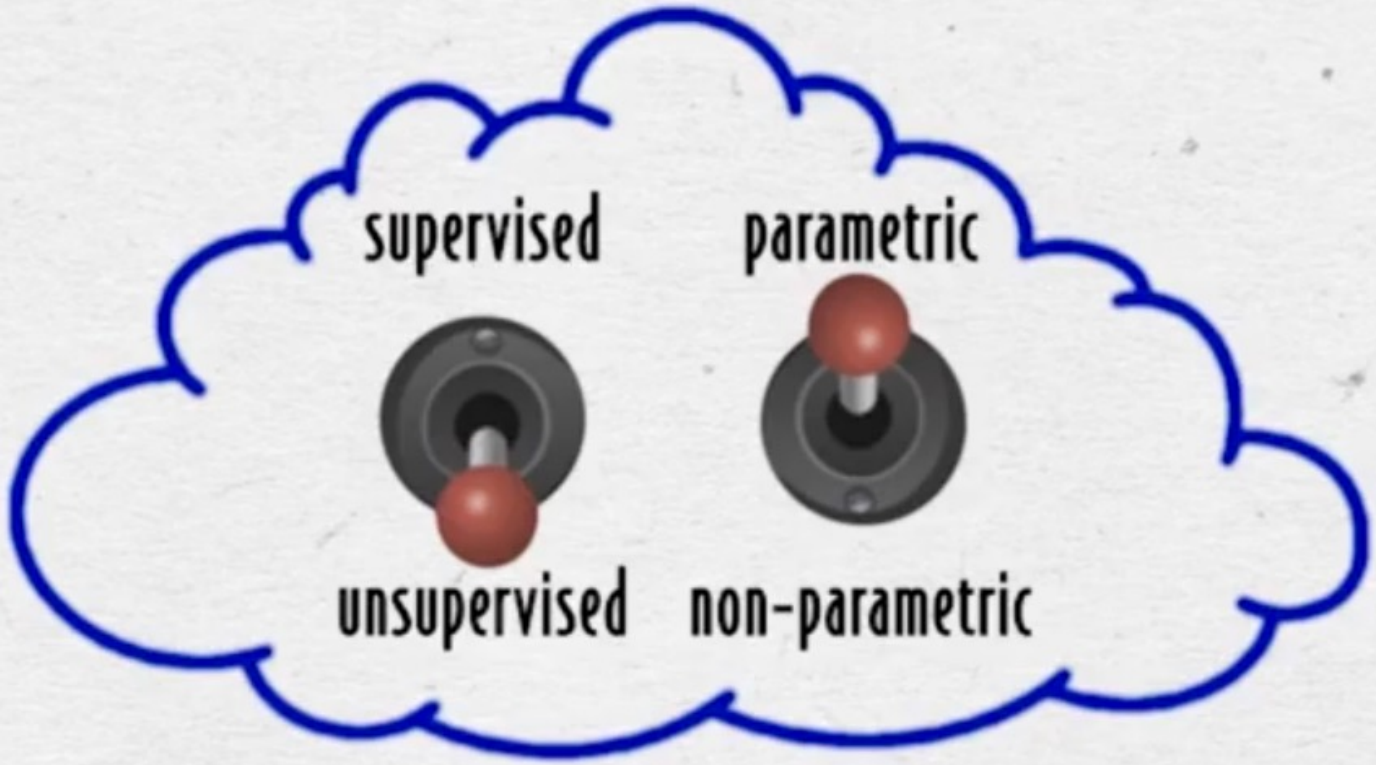
Parametric vs. Non-Parametric Learning

parametric: trial and error learning

non-parametric: counting and probablility


supervised
parametric
unsupervised
non-parametric
supervision: type of pattern being learned
parametricism: way the learning is stored

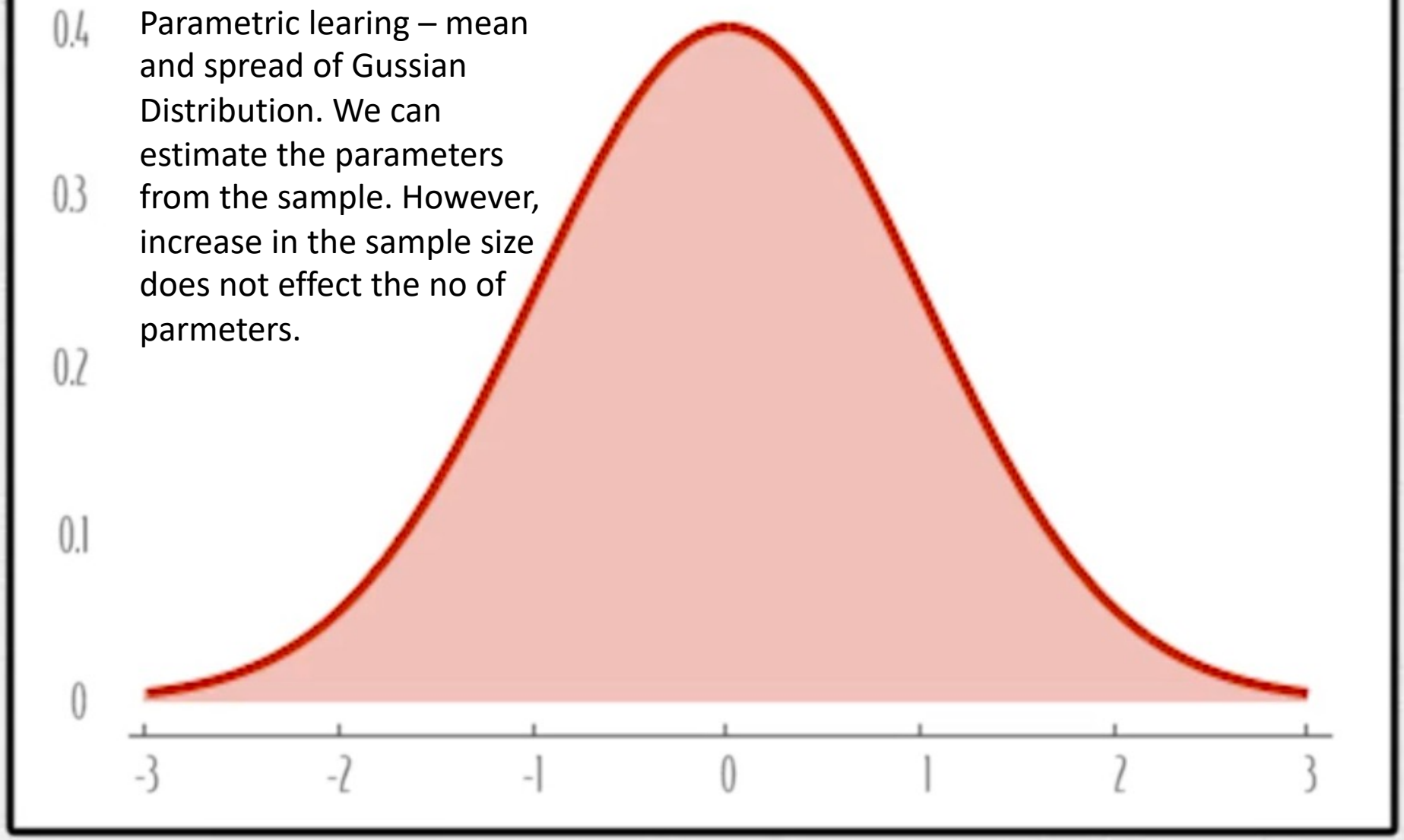

supervised
parametric
unsupervised
non-parametric



parametric - fixed number of parameters

non-parametric - possibly infinite parameters

Parametric learing – mean and spread of Gussian Distribution. We can estimate the parameters from the sample. However, increase in the sample size does not effect the no of parmeters.

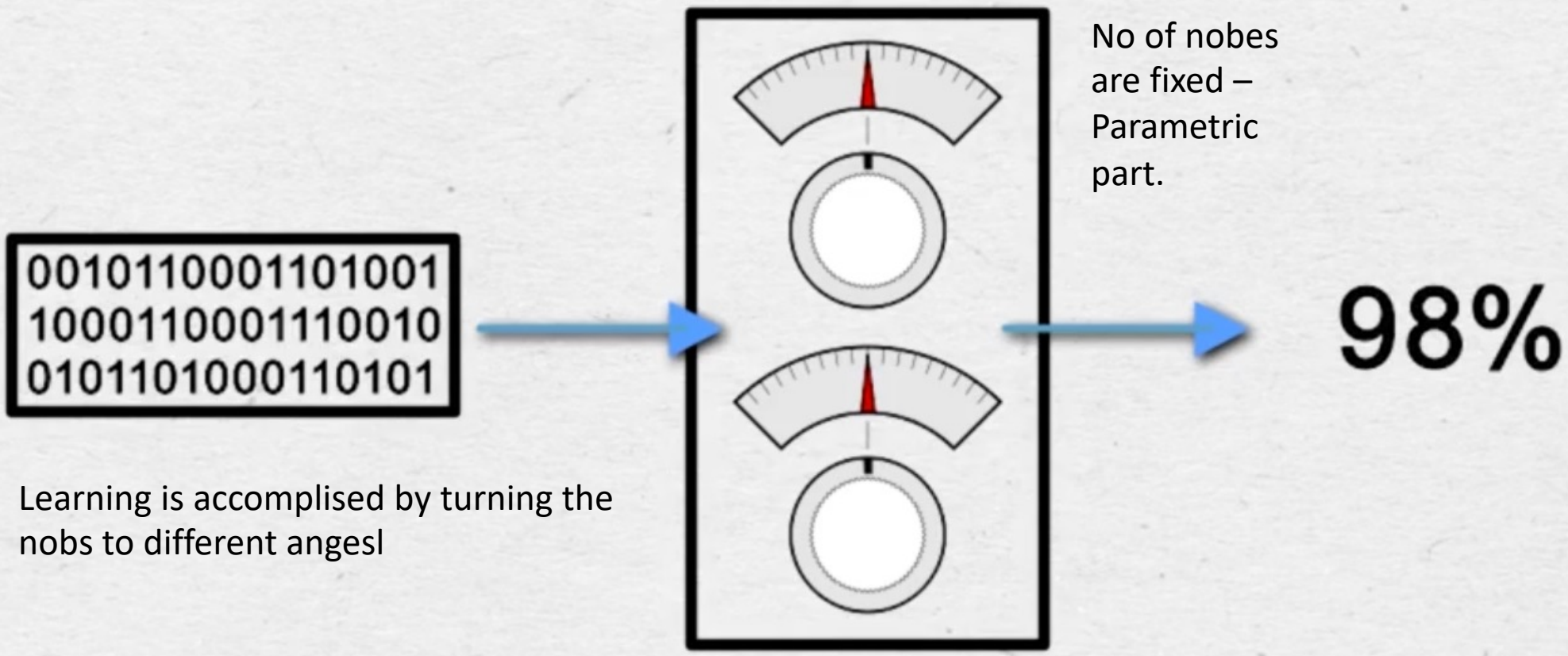


Supervised Parametric Learning

Data

Machine

Prediction



No of nobes are fixed – Parametric part.

Learning is accomplised by turning the nobs to different angesl

Supervised Parametric Learning

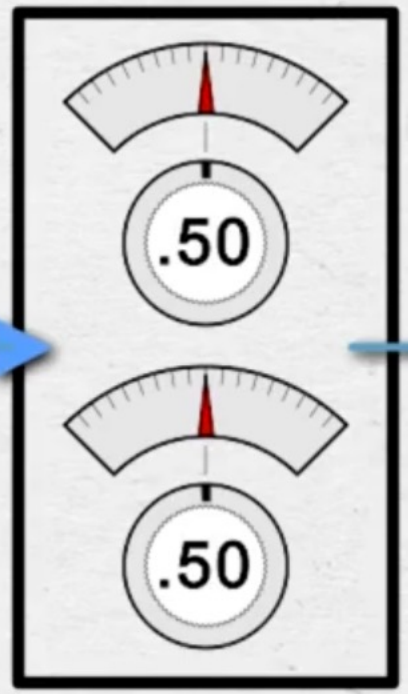
Data

Machine

Prediction

win/loss record

number of toes

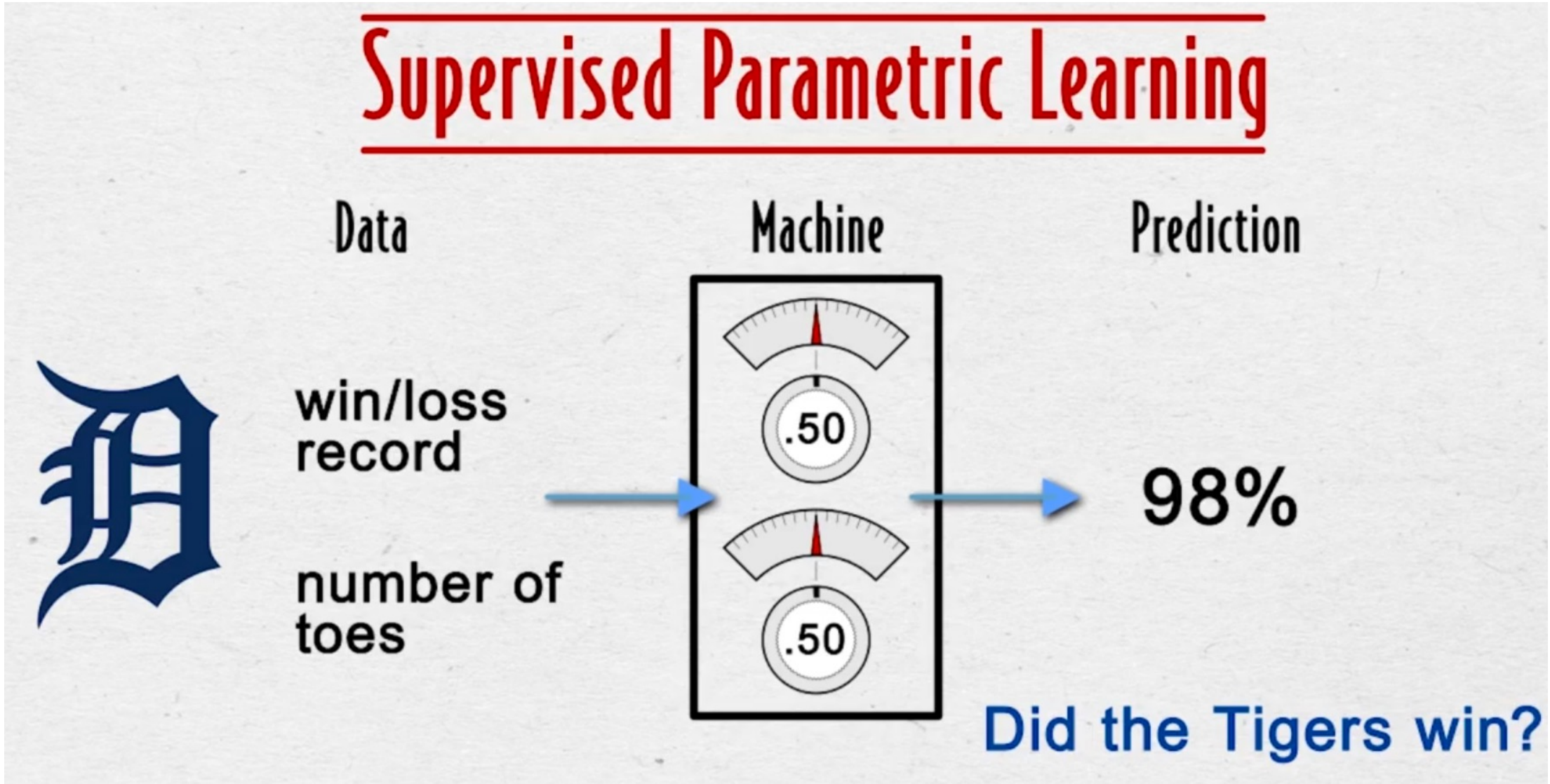


Learning is accomplished by turning the knobs to different angles.

98%

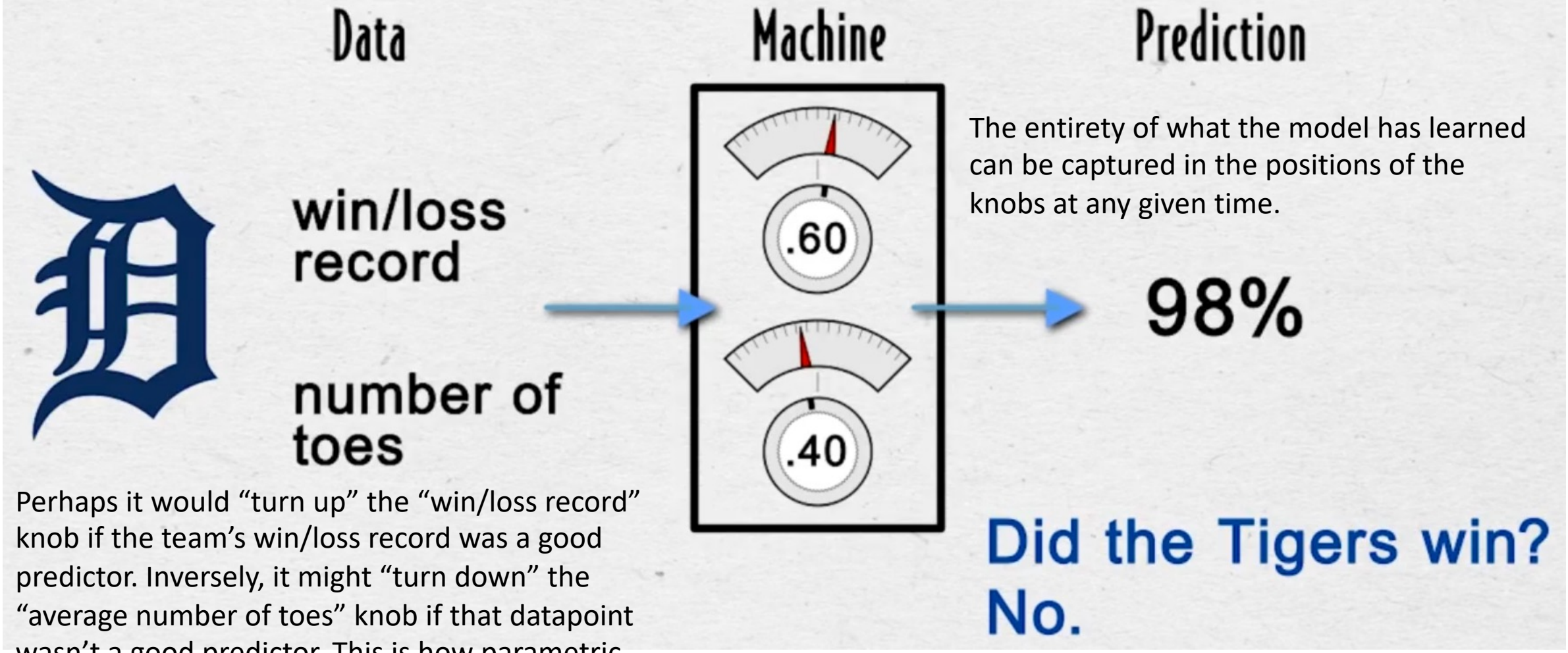
If we want to find Probabiity that Tigers will the World Series wins

Did the Tigers win?
No.



Next, the model would observe whether or not the Tigers actually won. After it knew whether they won, the learning algorithm would *update the knobs* to make a more accurate prediction the next time it sees the *same or similar input data*.

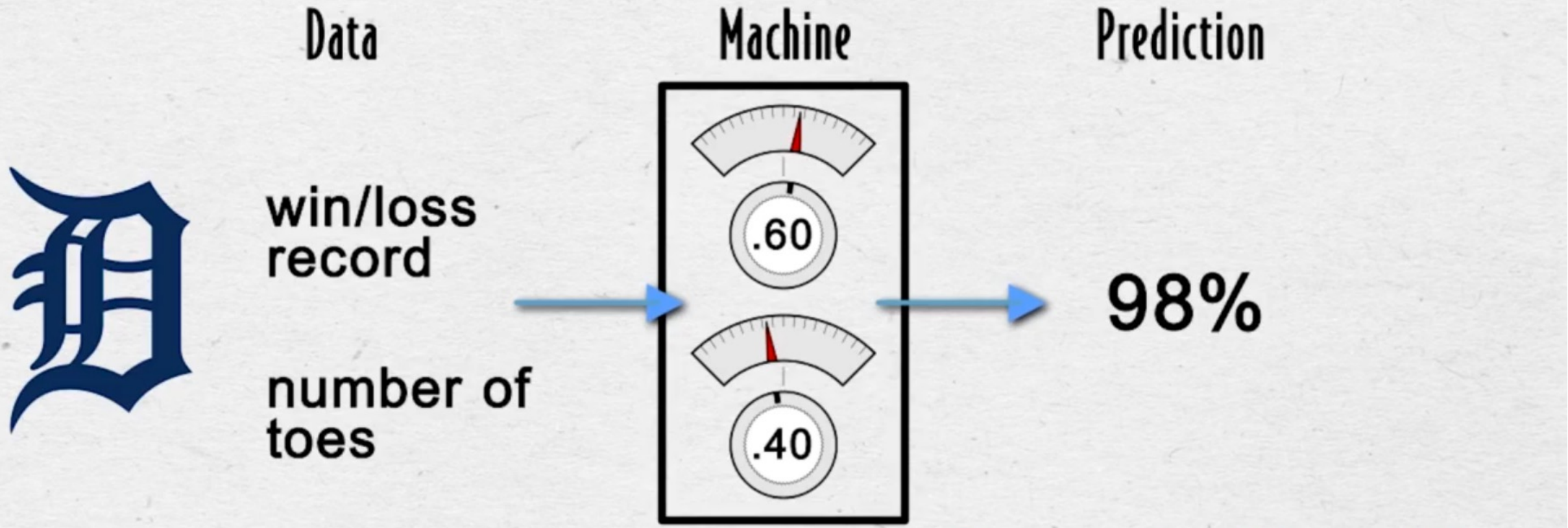
Supervised Parametric Learning



The entirety of what the model has learned can be captured in the positions of the knobs at any given time.

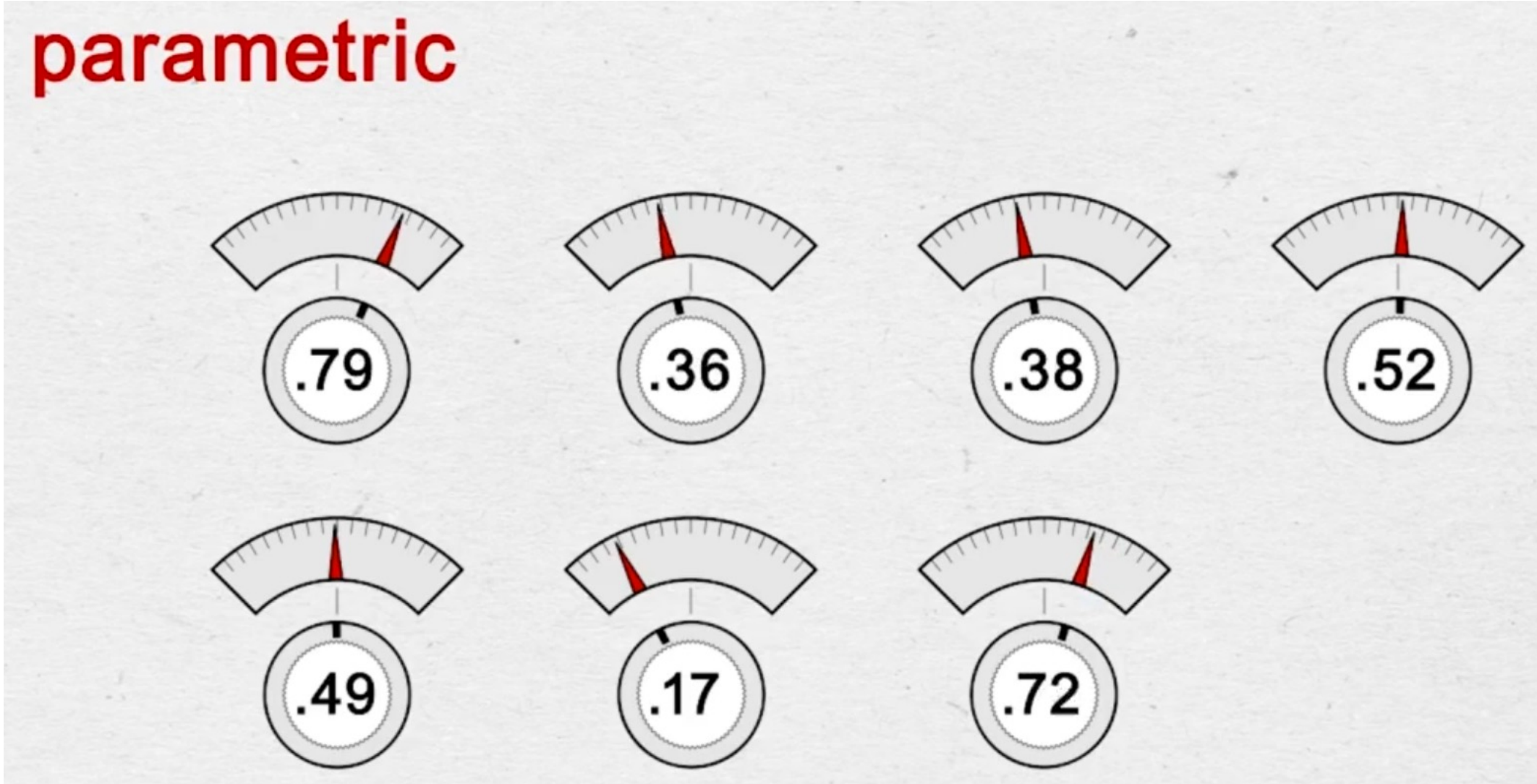
Perhaps it would “turn up” the “win/loss record” knob if the team’s win/loss record was a good predictor. Inversely, it might “turn down” the “average number of toes” knob if that datapoint wasn’t a good predictor. This is how parametric models learn!

Supervised Parametric Learning

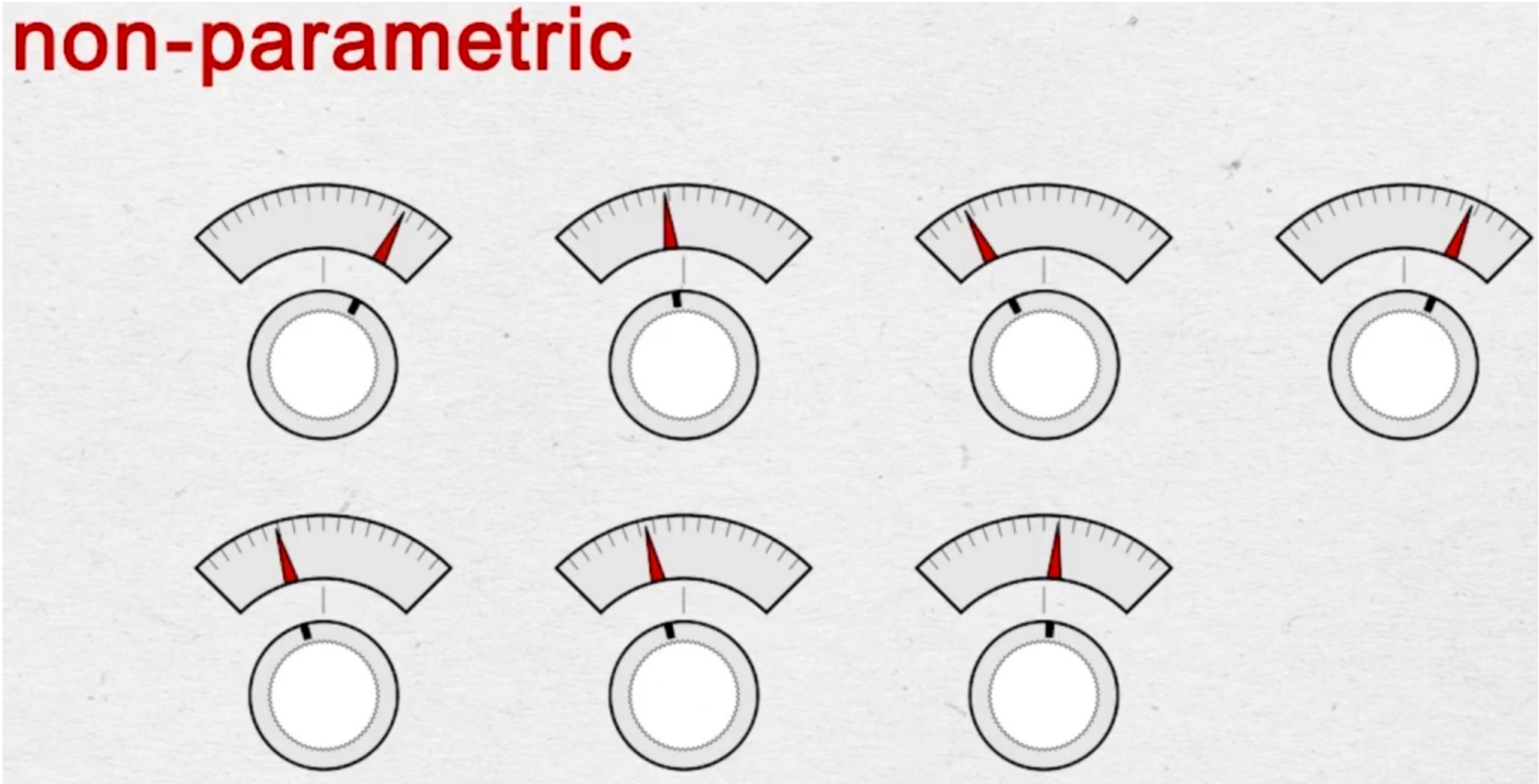


We can also think of this type of learning model as a search algorithm. We're "searching" for the appropriate knob configuration by trying configurations, adjusting them, and retrying.

Search Algorithm



Note further that the notion of trial and error isn't the formal definition, but it's a common (with exceptions) property to parametric models. When there is an arbitrary (but fixed) number of knobs to turn, some level of searching is required to find the optimal configuration.



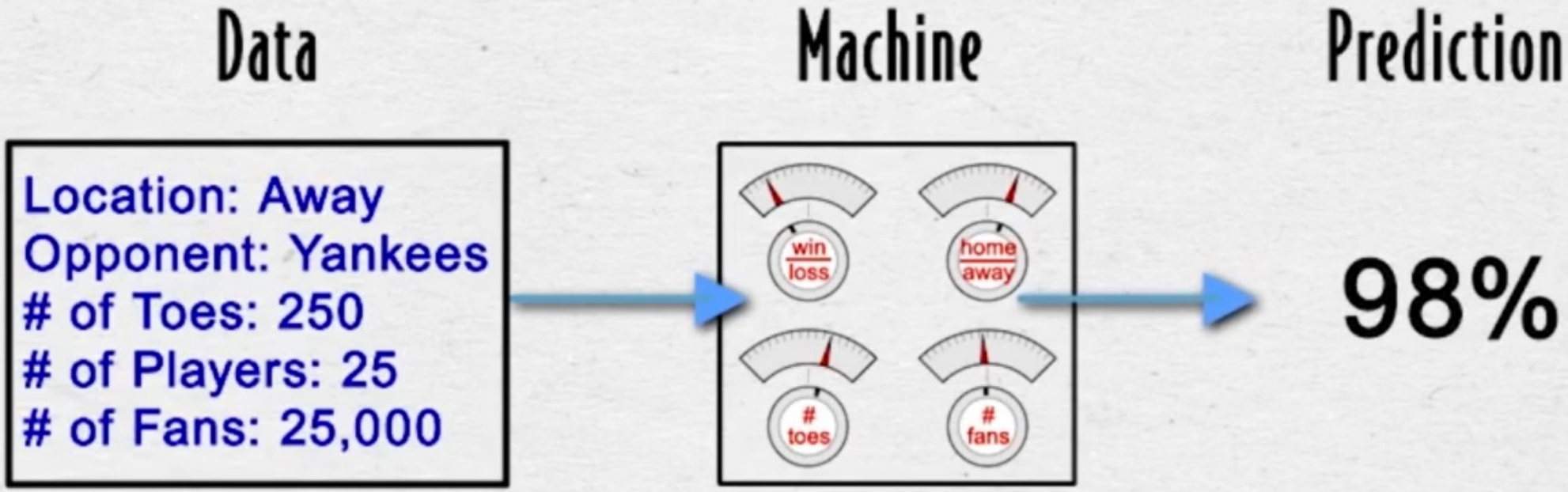
Nonparametric learning, which is often count based and (more or less) adds new knobs when it finds something new to count.

Three steps of Supversied Parametric Learning

Supervised Parametric Learning

Step 1: Predict

Let's continue with the sports analogy of trying to predict whether the Tigers will win the World Series.



Supervised Parametric Learning

Step 1: Predict

Step 2: Compare to Truth Pattern

Prediction: 98% > Truth: 0%

This step recognizes that if the model had predicted 0%, it would have perfectly predicted the upcoming loss of the team. You want the machine to be accurate, which leads to step 3.

Supervised Parametric Learning

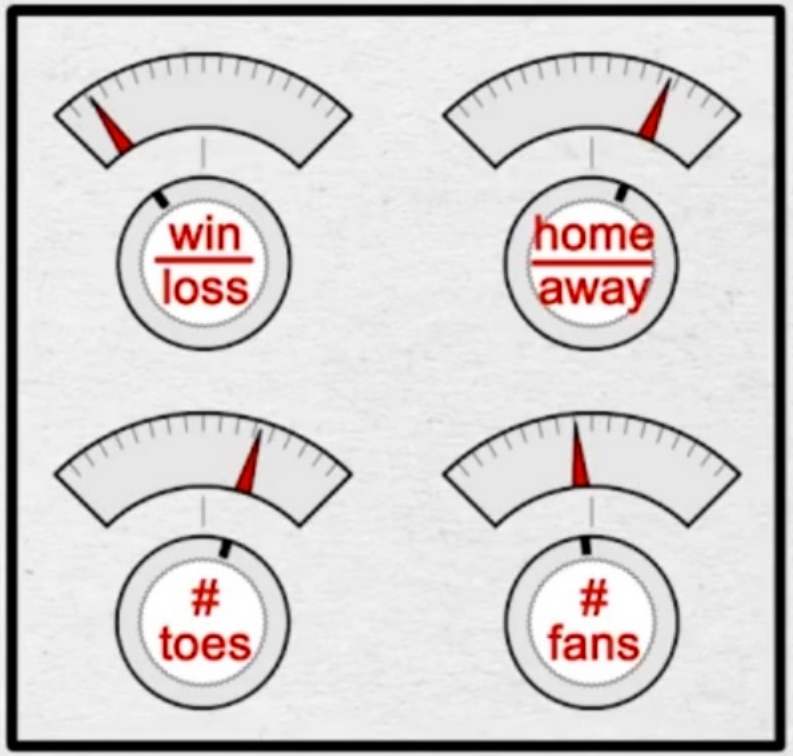
Step 1: Predict

Step 2: Compare to Truth Pattern

Step 3: Learn the Pattern

This step adjusts the knobs by studying both how *much* the model missed by (98%) and what the input data *was* (sports stats) at the time of prediction.

Adjusting Sensitivity by Turning Knobs



Supervised Parametric Learning

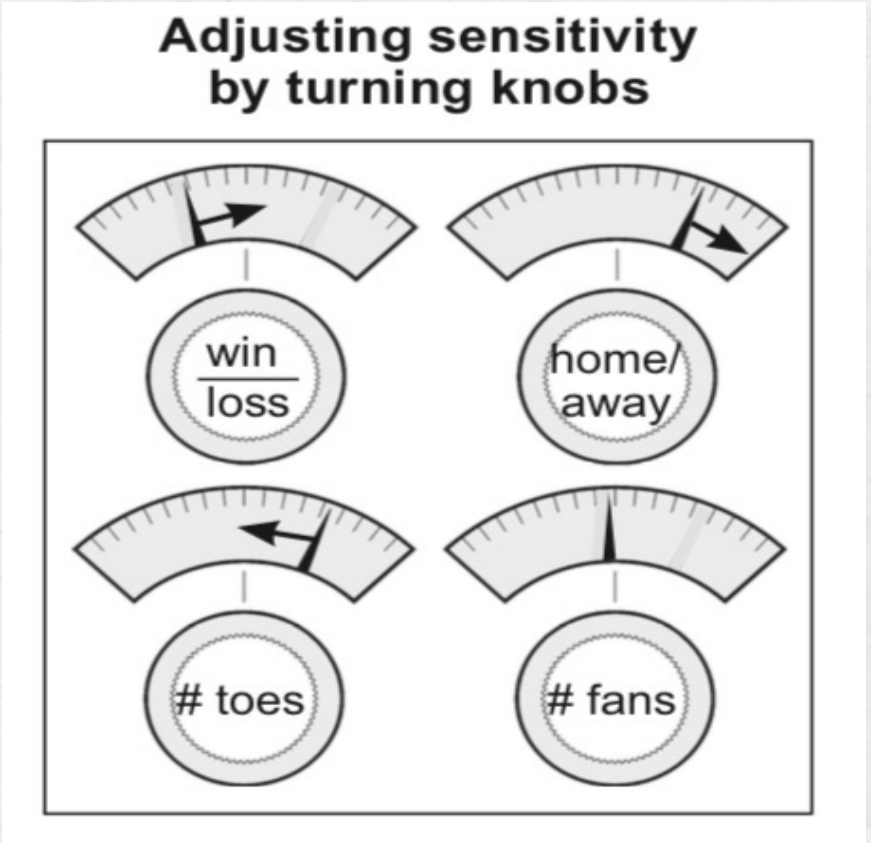
Step 1: Predict

Step 2: Compare to Truth Pattern

Step 3: Learn the Pattern

This step then turns the knobs to make a more accurate prediction given the input data.

Adjusting Sensitivity by Turning Knobs



Supervised Parametric Learning

Step 1: Predict

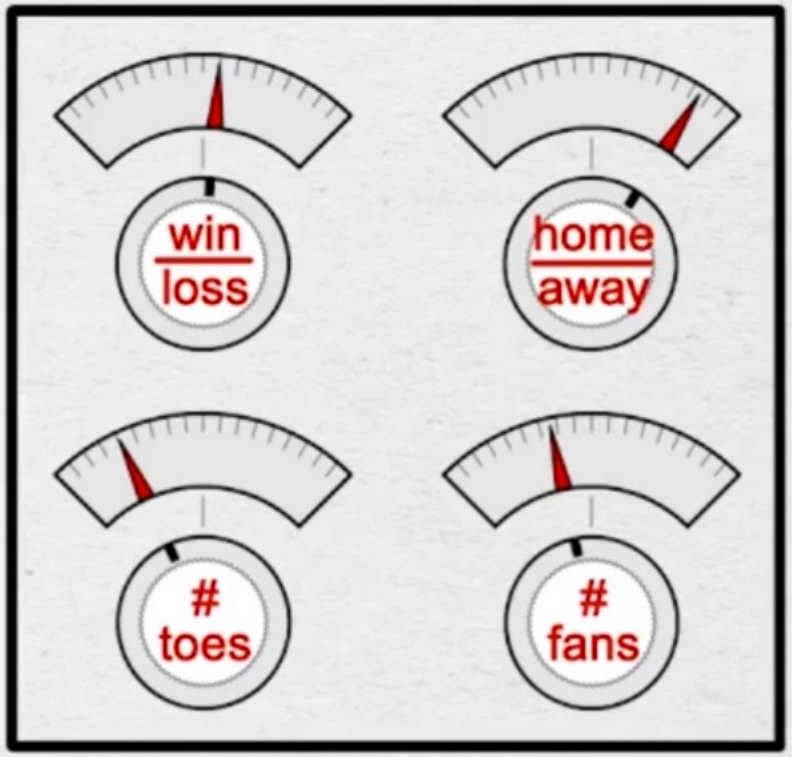
Step 2: Compare to Truth Pattern

Step 3: Learn the Pattern

The next time this step saw the same sports stats, the prediction would be lower than 98%.

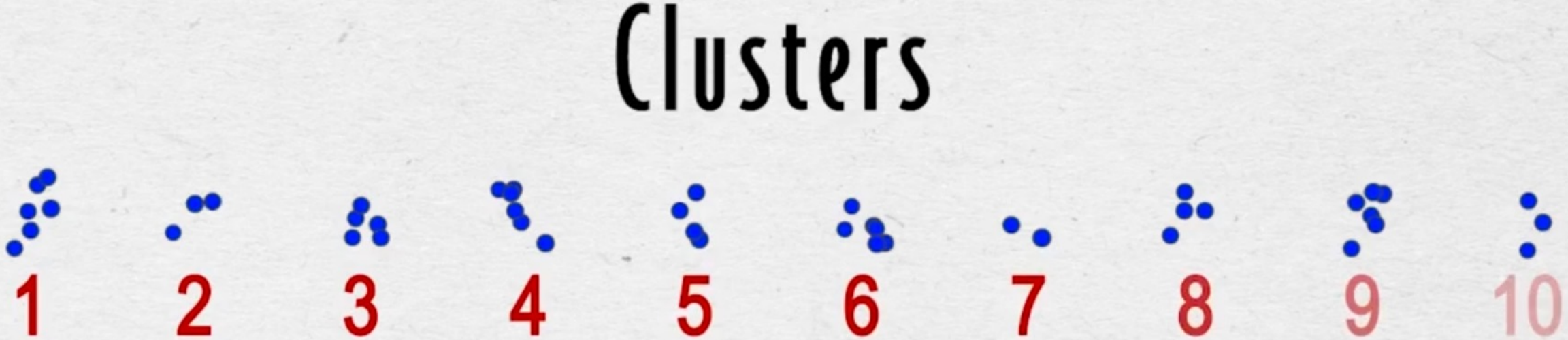
Note that each knob represents the *prediction's sensitivity to different types of input data*. That's what you're changing when you "learn."

Adjusting Sensitivity by Turning Knobs



Unsupervised Parametric Learning

Unsupervised learning is all about grouping data. Unsupervised *parametric* learning uses knobs to group data. But in this case, it usually has **several knobs for each group**, each of which maps the input data's affinity to that particular group



Unsupervised Parametric Learning

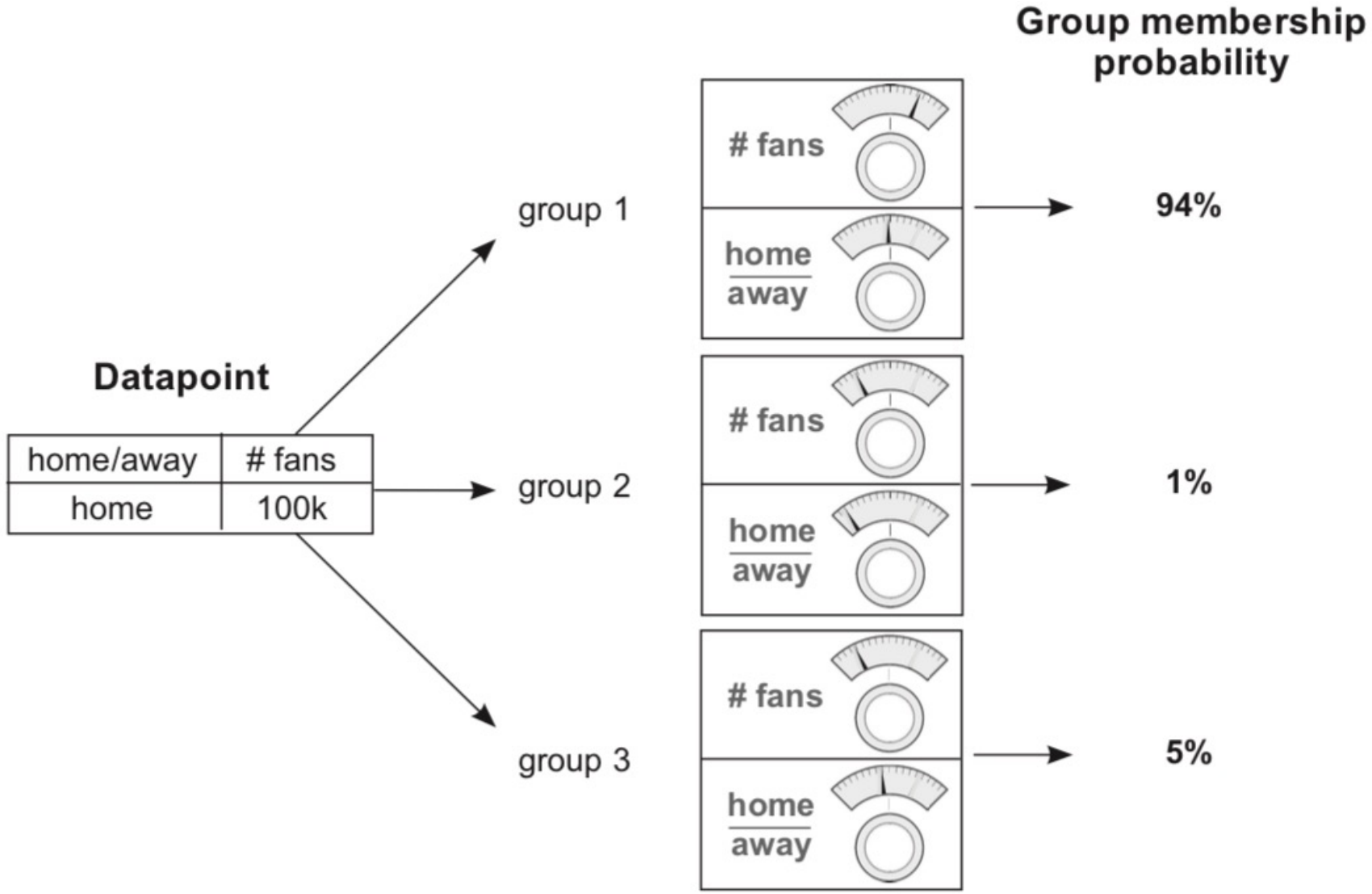
There are three clusters in the data

home/away	#fans
home	100k
away	50k
home	100k
home	99k
away	50k
away	10k
away	11k

Datapoint

home/away	#fans
home	100k

Group Membership Probability


Group membership probability
Datapoint
home/away
fans
home
100k
group 1
group 2
group 3
fans
home
away
94%
1%
5%

Unsupervised Parametric Learning

home/away	#fans
home	100k
away	50k
home	100k
home	99k
away	50k
away	10k
away	11k

Each group's machine attempts to transform the input data to a number between 0 and 1, telling us the *probability that the input data is a member of that group*.

Datapoint

home/away	#fans
home	100k

There is a great deal of variety in how these models train and their resulting properties, but at a high level they adjust parameters to transform the input data into its subscribing group(s).

Group Membership Probability

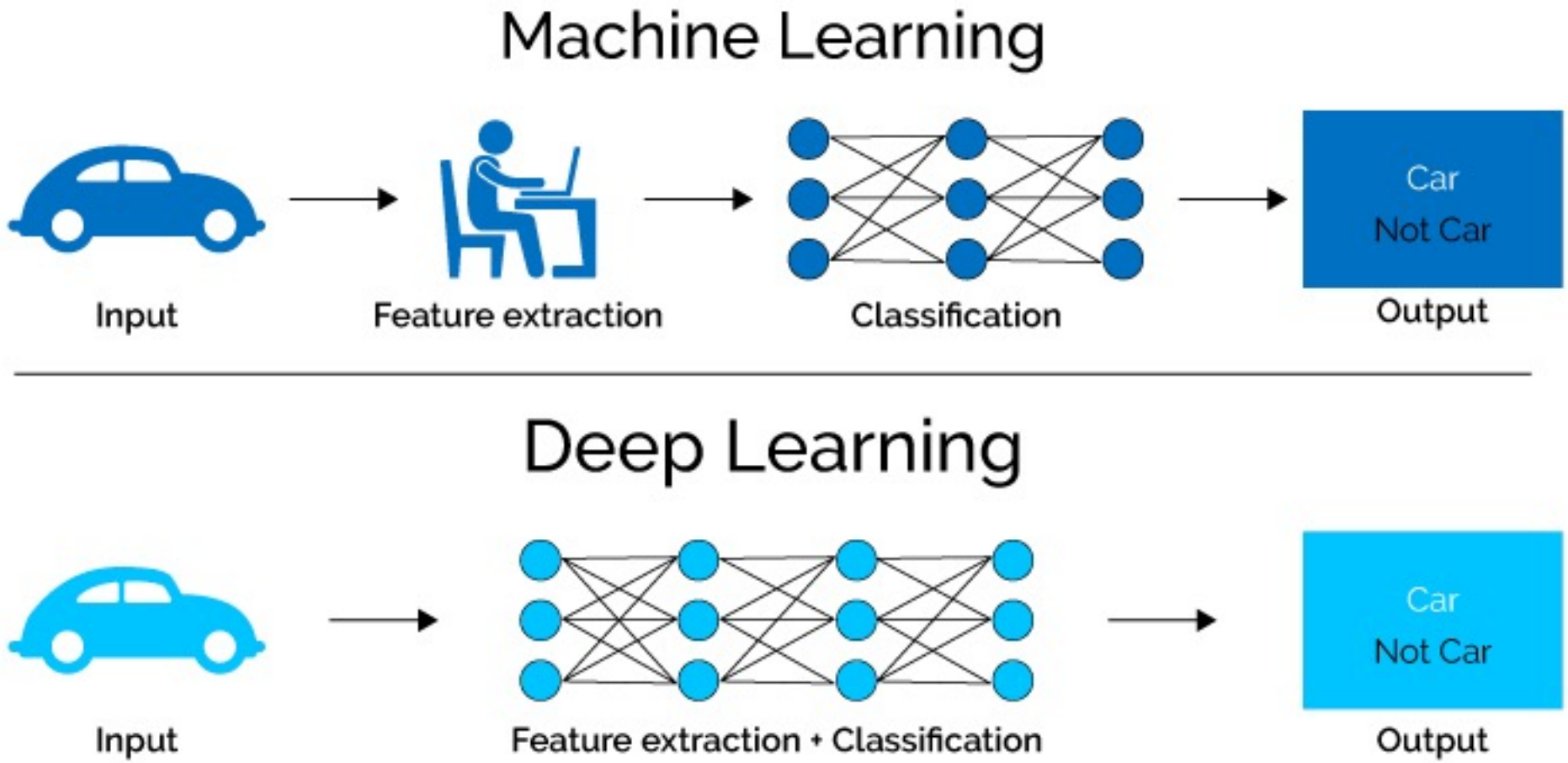
94%

1%

5%

Non-Parametric Learning

parameters based on data


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output